RL-TR-92-345, Vol I (of seven)
Final Technical Report
December 1992

SYSTEM ENGINEERING CONCEPT DEMONSTRATION, Effort Summary

Software Productivity Solutions, Inc.

Edward R. Comer, Sharon L. Rohde

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-92-345, Volume I (of seven) has been reviewed and is approved for publication.

APPROVED: Frank S LaMonica

FRANK S. LAMONICA
Project Engineer

FOR THE COMMANDER:

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)

2. REPORT DATE
December 1992

3. REPORT TYPE AND DATES COVERED
Final Feb 90 - Jul 92

4. TITLE AND SUBTITLE
SYSTEM ENGINEERING CONCEPT DEMONSTRATION,
Effort Summary

5. FUNDING NUMBERS
C - F30602-90-C-0021
PE - 62702F
PR - 5581
TA - 18
WU - 54

6. AUTHOR(S)
Edward R. Comer
Sharon L. Rohde

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Productivity Solutions, Inc.
122 4th Avenue
Indialantic FL 32903-1697

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CB)
525 Brooks Road
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-92-345 Vol I
(of seven)

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Frank S. LaMonica/(315) 330-2054

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This final technical report documents the objectives, activities, and results of Air Force contract F30602-90-C-0021, entitled "System Engineering Concept Demonstration." The effort, which was conducted by Software Productivity Solutions, Inc., with McDonnell Douglas Corporation - Douglas Aircraft Company and MTM Engineering Inc. as subcontractors, demonstrated and documented the concept of an advanced computer-based environment which provides automation for Systems Engineering tasks and activities within the Air Force computer-based systems life cycle. The report consists of seven (7) volumes as follows: I) Effort Summary, II) Systems Engineering Needs, III) Process Model, IV) Interface Standards Studies, V) Technology Assessments, VI) Trade Studies, and VII) Security Study.

This Volume (Volume I - Effort Summary), defines the scope of the effort, describes the contractual activities that were performed, and summarizes the results contained in companion Volumes II - VII.

14. SUBJECT TERMS
System Engineering, System Life Cycle Tools, System Life Cycle Environment

15. NUMBER OF PAGES
122

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

Executive Summary

The objective of the System Engineering Concept Demonstration (SECD), contract F30602-90-C-0021 for the Air Force Rome Laboratory (RL), was to demonstrate the concept of an advanced computer-based environment to support Air Force computer-based systems life cycle. This advanced computer-based environment seeks to increase the productivity and effectiveness of systems and specialty engineers involved in the development, maintenance, and enhancement of military computer-based systems.

To meet this objective, the Systems Engineering Concept Demonstration effort was scoped to address the systems engineering process and the automation of the systems engineering role. The SECD Project Team focused on high payoff automation (e.g., modeling and specification meta-tools, concurrent engineering groupware, integration mechanisms, environment administration support) and generic systems engineering functions that could be interfaced and adapted to specific applications, levels, and organizations.

The SECD project tasks included analysis of systems engineering needs, development of a generic process model, study of emerging interface standards, technology assessments, trade studies, and a security study. Prototype concept and technology demonstrations were developed which focused on risk abatement to provide evidence of feasibility for potential users.

As an outgrowth of this research, a System/Segment Specification, a System Specification Design Document and an Interface Requirements Specification were produced, all compliant with MIL-STD-2167A. Additionally, an Operational Concept Document, and a Rationale Document for requirements and design decisions were produced to clearly define the concept of automating system engineering and capture informal information for future development.

The name *Catalyst* was selected to refer to the systems engineering automation objective of this effort - reflecting a view that the systems engineer and his/her associated systems engineering automation serve as the "catalyst" for the various specialty roles required in developing a system. Just as a chemically-based catalyst is a substance that modifies and increases the rate of a chemical reaction, Catalyst enhances the capabilities of systems and specialty engineers and provide an effective environment for increased and highly efficient interaction between them.

Catalyst will provide an automated environment of integrated, state-of-the-art software tools and methods which support systems engineering throughout the life cycle. The feasibility, usability and marketability of Catalyst has been fortified by the various trade-off studies and analyses, and concept and technology demonstrations that SPS performed.

1. Introduction

This Final Technical Report (FTR) documents the objectives, activities, and results of the Systems Engineering Concept Demonstration, contract F30602-90-C-0021 sponsored by the Air Force Systems Command (AFSC) Rome Laboratory (RL). The effort was conducted by Software Productivity Solutions, Inc. (SPS) with the McDonnell Douglas Corporation Douglas Aircraft Company (MDC-DAC) and MTM Engineering Inc. as subcontractors.

As additional tasks of the SECD effort, a System/Segment Specification (SSS), System/Segment Design Document (SSDD), and Interface Requirements Specification (IRS) were produced in accordance with DoD-STD-2167A data item descriptions. An Operational Concept Document (OCD) was produced in accordance with the American Institute of Aeronautics and Astronautics (AIAA) Recommended Technical Practice. The fundamental reasoning and basis for requirements, design, and interfaces was also captured in a System Specification Rationale (SSR) document. This FTR does not not duplicate information contained in these DoD-STD-2167A documents/specifications, but instead serves as a companion document, providing a summary of the SECD effort and insight and rationale into the resulting system concept. All documents for the SECD effort were finalized July 22, 1992.

1.1. Organization of the Document

The SECD FTR provides a summary of the effort and has six supporting volumes associated with it:

Volume 1 - Effort Summary

Volume 2 - Systems Engineering Needs

Volume 3 - Process Model

Volume 4 - Interface Standards Studies

Volume 5 - Technology Assessments

Volume 6 - Trade Studies

Volume 7 - Security Study

Volume 1, Effort Summary, includes the effort and scope of SECD; summaries of system engineering needs, the process model, interface standards studies, technology assessments, trade studies, a security study; and a discussion of the SECD Project Team's participation in the National Council on System

Engineering (NCOSE). The conclusions of the entire SECD effort and areas for future Research and Development are also discussed in Volume 1.

The elaborated and final work products of each of these areas are found in their associated volumes making the FTR a summary of the extensive research effort that ensued. The conclusions of each of these volumes appear in Volume 1 as well as in the corresponding separate volume to provide completeness and continuity. A list of references is provided at the end of each volume.

1.2. Background

The engineering challenge of today is to find ways of building bigger and better structures, equipment, and systems with the available resources. Cost, schedule, and available-resources constraints (e.g., hardware, operating systems, development personnel) provide all engineering disciplines with a constant challenge. Systems engineering is particularly challenging due to the quick growth in technology, the breadth of knowledge required, the ever-expanding demands for more and more capabilities in smaller and smaller packages, changes in customer requirements, and demands for more reliable, more maintainable, high-performance, high-quality systems.

The importance of the systems engineering process has only been recently highlighted. For example, a 1984 CODSIA report [COD84] highlighted the need for a 'strong **systems** technology initiative.' A more recent investigation by the House of Representatives [REP89] on computer software development identified a 'systems-first' approach as being key to solving a number of computer systems problems:

> Critical to the new procurement statute must be a 'system first' perspective. It is important that consideration for system requirements drive managers. Allowing hardware and software development to proceed separately or in isolation is a formula for problems. Giving software equal status in planning for procurement will certainly change Government procurement. Good systems engineering, where the program manager factors in user needs, safety and security at the outset of design and seeks tradeoffs to match his available resources, may leave nothing tangible to show when the time comes for the next procurement cycle. No program manager relishes the thought of defending a request for funds when the major activity seems to be endless arguments over obtuse technical points by large numbers of well-paid engineers. Yet experience shows that this is precisely the course to follow because it answers most, if not all, the questions that are expensive to fix on a production line. [REP89]

But system engineering is hard! It is not formalized, there are few textbooks, many processes are fuzzy, and it covers a broad range of engineering and management disciplines. System engineering involves trade-off analyses that require the development of many alternative architectures, engineering, and manufacturing scenarios. Techniques for comparing different approaches range from abstract assessments of resultant system quality and engineering and manufacturing risks to objective resource , feasibility and schedule analysis. These activities vary over a range of informal reasoning and decision making to formal models of systems.

System engineering is itself the integration of many engineering disciplines. A critical problem in most organizations is that system engineering is predominantly lopsided; there is a heavy emphasis in one engineering discipline, usually hardware. This is typically a consequence of the history of the organization. More forward looking organizations that are attempting to combine the hardware and software disciplines during system engineering are faced with the problem of widely disparate toolsets and environments. It is difficult to effectively combine the functions of different engineering disciplines to form a true system engineering environment.

1.3. Objectives of the Effort

The objective of the Systems Engineering Concept Demonstration, as specified in the statement of work, was as follows:

> "The objective of this effort is to demonstrate the concept of an advanced computer-based environment of integrated software tools and methods which supports the Air Force computer-based systems (i.e., software, firmware, and hardware) life cycle - short of actual hardware fabrication. The intent of the environment is to provide integrated, state-of-the-art engineering and development capabilities throughout the complete system life cycle. System requirements/design and software requirements of the system engineering and development environment shall be developed and documented. Demonstrations of technologies which are critical to establishing a system engineering and development environment capability shall be provided." SOW 1.1

By providing integrated, state-of-the-art engineering and development capabilities throughout the complete system life cycle, systems and specialty engineers can increase their productivity and effectiveness during the development, maintenance, and enhancement of military computer-based systems.

1.4. Scope of the effort

Systems engineering is a very broad topic: there are numerous different types of systems and there are numerous engineering, analysis, assurance and management disciplines that are involved in developing these systems. Questions such as "what is a system" and "what is systems engineering" are not answered with any general consensus.

Thus, there was a significant risk in addressing automation of systems engineering with a scope that is too broad or too ill-define will seriously impede the effort. In order to better define and narrow the scope of the effort to a point of feasibility, the following scoping decisions were made:

1. The effort addresses the *systems engineering process* and not the entire *engineering of systems process*. The engineering of systems involves numerous specialty engineering, analysis, assurance and management roles that span the entire life cycle. Specialty engineers include subsystem engineers (e.g., hardware, software, electrical, mechanical, and structural); analysts (e.g., mission analysis, reliability, human factors, producibility, and supportability); and technology specialists (e.g., sensors, electronic warfare, weapons). Systems engineering is one role that serves as the glue or catalyst between these various disciplines.
2. The effort addresses primarily the automation of the systems engineering role. For the various other specialty roles, only those activities that support the systems engineering process will be addressed.
3. The effort focuses on *high payoff* automation, rather than attempting to provide comprehensive coverage for all of the systems engineering automation throughout the life cycle.
4. The effort seeks to automate the more *generic* systems engineering functions (e.g., alternatives analysis, tradeoffs, requirements elaboration and allocation) and to either exclude (and interface to) or adapt to areas that are specific to the type of application, the level of system being engineered, the organization or the methods being applied.

The Systems Engineering Concept Demonstration is one of the first efforts to seriously address automation of the systems engineering process. As such, the SECD Project Team identified these areas as the most appropriate for the scope of this effort.

The name *Catalyst* was selected to refer to the systems engineering environment objective of this effort, reflecting our view that the systems engineer, and therefore the systems engineering automation, is the catalyst for the various specialty roles required in developing systems.

1.5 Summary of Project History

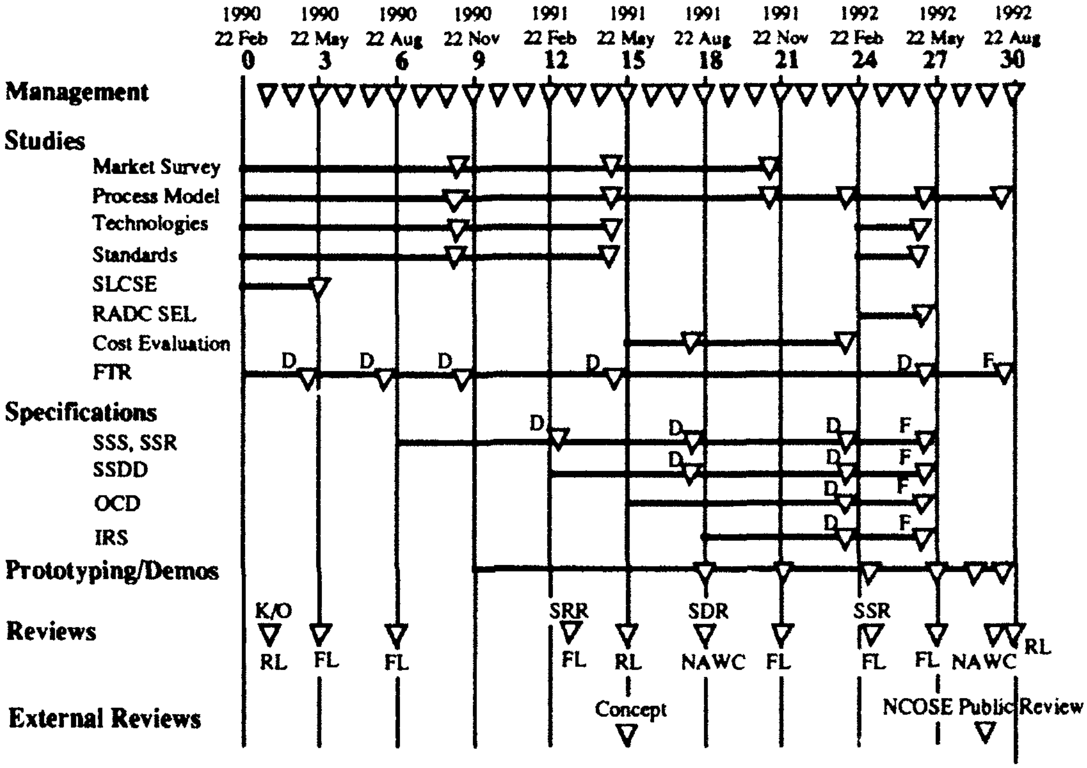
1.5.1 Tasks

The tasks of SECD were divided into the following areas:

1. Management
2. Studies
   a) Market Survey
   b) Process Model
   c) Technologies
   d) Standards
   e) SLCSE Evaluation
   f) Rome Laboratory Software Engineering Facility Evaluation
   g) Cost Evaluation
   h) Final Technical Report (FTR)
3. Specifications
   a) System Specification (SSS)
   b) System Design Document (SSDD)
   c) Operational Concept Dcoument (OCD)
   d) Interface Requirements Specifification (IRS)
4. Prototyping and Demosntrations
   a) Tradeoff
   b) Timeline
   c) Flowdown
   d) Technology Demonstrations
5. Reviews
   a) Quarterly Reviews
   b) System Requirements Review (SRR)
   c) System Design Review (SDR)
   d) Software Requirements Review (SRR)
6. External Reviews
   a) Concept presentation
   b) Public review at NCOSE conference

1.5.2 Schedule

The following schedule defined the timetable for the SECD tasks, deliverables and reviews.



Legend

D Draft
F Final
K/O Kickoff
RL Rome Laboratory
NAWC Naval Air Warfare Center, Warminster
FL Florida - SPS, Indialantic

Figure 1.5.2-1 SECD Program Schedule

2. Summary of Systems Engineering Needs

Volume 2 of this FTR, entitled Systems Engineering Needs, describes the various investigations that were conducted to analyze and prioritize the systems engineering needs that Catalyst will satisfy, as follows:

1. Needs Survey of several references that investigated the problems with mission-critical systems development and maintenance.
2. Field Interviews of practicing systems engineers:
   a) conducted under the SECD effort at selected government and contractor facilities in 1990; and
   b) conducted previously by the Microelectronics and Computer Technology Corporation (MCC) at member companies in 1986.

2.1 User Profile

The envisioned automated systems engineering product is targeted toward large-scale developers and maintainers of mission-critical systems. Systems engineering is conducted at many levels in a complex program structure involving numerous organizations and groups. A migration to concurrent engineering approaches is being driven by increasing system complexity that results in the systems engineering function being more dependent on the participation of specialty and subsystem engineering organizations. Table 2.1-1 provides an organizational profile for the target users.

The automated environment targeted the systems engineering *process,* that involves systems engineers interacting with a variety of other specialty roles. Therefore, the users of the environment included not only the systems engineers, but also other related and supporting specialty roles. However, the intention of the environment was not to automate all activities of these supporting roles, but to augment their own specialty automation with meaningful tools that support the systems engineer and the systems engineering process.

Thus, the systems engineers were considered the primary users of the *Catalyst* environment. This environment is likely, in the near term, to provide the majority of the automation that the systems engineer utilizes on a regular basis throughout the life cycle. Table 2.1-2 provides a profile of the typical systems engineer.

The other specialty engineering, analysis and management roles were considered secondary users of the *Catalyst* environment. Their use of the *Catalyst* tools evolved around activities that support and interface with the systems engineering process. However, the majority of their specialty role was assumed to be automated by other tools. Table 2.1-3 contrasts the characteristics of the specialty role users.

Table 2.1-1. Organizational Profile

Characteristic	Organization Profile
Organizational composition	Multi-organization collaboration consisting of multiple customer (government) and developer (contractor) organizations.
Geography	Widely geographically distributed (airline travel required to assemble all of the players).
Program organization	Hierarchical program organization consisting of multi-layer contractor-subcontractor relationships (includes internal contracts). An organization may have an underlying matrix structure that supports multiple programs.
Standards and policies	Diverse collection of Government and organizational standards, policies and conventions all various levels.
Computing resources	Heterogeneous collection of mainframes, workstations and personal computers. The workstations run Unix, while the mainframes and personal computers may use other operating systems.
Local area networking	Incomplete collection of local area networks that may or may not include gateways. Complete connectivity between all resources cannot be relied upon.
Wide area networking	Some point-to-point connections and access to public and government wide area networks that provide basic electronic mail and file transfer capabilities. Complete electronic connectivity between all sites cannot be relied upon. Frequent use of fax.
Engineering software	Diverse collection of COTS and organization-developed tools. Much of the software is unique to a class of computing platform. Software is not uniformly distributed across platforms.
Software integration	Some loosely integrated tools primarily using data interchange standards and tool-to-tool conversions.
Engineering techniques	Few standards and guidelines span organizations. Wide variance in techniques across organizations. Some local organizational standard techniques and conventions.

Table 2.1-2. Individual Systems Engineer Profile

Characteristic	Individual Profile
Education	BS in a technical field
Years of experience	10 - 20 years
Job history	Initial jobs in specialty or subsystem engineering positions with increasing responsibilities. Migration into systems-level responsibilities.
Knowledge	Considerable experience in the application domain (possibly the entire career) with deep application domain knowledge. Broad, but not deep, knowledge of a number of speciality engineering areas.
Computer usage	Infrequent to moderate computer user, primarily on a personal computer.
Computer literacy	Slightly to moderately proficient with general purpose tools and a small set of specialty tools that are used a lot. Little or no formal training or education in computer or tool usage.
Communication skills	Good, persuasive technical communication skills.

Table 2.1-3. Individual Supporting Specialty Role Profile

Characteristic	Individual Profile
Education	BS or MS in an engineering field
Years of experience	2 - 15 years
Job history	Increasing levels of responsibility in the specialty role.
Knowledge	Considerable experience and knowledge in the specialty engineering area. Somewhat familiar with the application domain. Very limited knowledge of other specialty areas.
Computer usage	Moderate to frequent computer user, involving not only personal computers, but also mainframes and workstations.
Computer literacy	Moderately to highly computer proficient.
Communication skills	Moderate communication skills.

2.2. Needs Summary

Systems engineering activities can be categorized into three types of activities:

1. Engineering
2. Communication
3. Management

The precise mixture of engineering, communication and management activities varies by individual and will often change over the life cycle. For example, the systems engineer may begin by performing primarily an engineering role, defining requirements as part of a small team. As the project expands into system design, large amounts of the systems engineers time may be spent coordinating the activities of the subsystem engineers and disseminating a common understanding of the requirements. Once the systems design is complete and the requirements allocated, the systems engineer may be largely performing a management role, planning and tracking the subsystem developments.

A total of 18 different activities are listed below that are frequently associated with the systems engineering process:

1. **Engineering**
   - 1.1 Requirements Engineering
   - 1.2 System Design & Allocation
   - 1.3 Interface Definition & Integration
   - 1.4 Tradeoff Analysis
   - 1.5 Engineering Decision Making
   - 1.6 Change Impact Analysis & Management
   - 1.7 Integration Planning and Management
   - 1.8 Quality Engineering and Assurance
   - 1.9 Specification Generation
2. **Communication**
   - 2.1 Collaboration & Coordination
   - 2.2 Information Research
   - 2.3 Boundary Spanning
   - 2.4 Joint Work Product Development

3. **Management**
    - 3.1 Standard and Policy Application
    - 3.2 Process Management
    - 3.3 Program Planning & Tracking
    - 3.4 Task Management
    - 3.5 Risk Analysis

Affecting each of these activities is the problem of electronic handling of classified information. There is a need for pragmatic solutions that allow a high degree of automated support when some or all of the information involved is classified.

The following subsections overview systems engineering needs for each of these activities.

2.2.1 Engineering Needs

This section discusses the engineering needs for the following activities:

- Requirements Engineering
- System Design & Allocation
- Interface Definition & Integration
- Tradeoff Analysis
- Engineering Decision Making
- Change Impact Analysis & Management
- Integration Planning and Management
- Quality Engineering and Assurance
- Specification Generation

Requirements Engineering. Because the requirements are as complex as the system they represent, the tasks of organizing requirements, precisely specifying requirements without ambiguity, and insuring consistency by removing conflicting or misleading requirements, are all extremely difficult. Moreover, the process of turning requirements into systems is a very dynamic task, involving significant iteration and change.

Most programs have difficulty defining requirements and dealing with constantly changing requirements. [RED84] As much as fifty-five (55) percent of system errors are introduced during the requirements definition process. It is believed that proper (i.e., systematic, disciplined) requirements engineering would alleviate this situation.

Yet, many in-practice requirements engineering methods are being applied manually or are not adequately supported by automation. Most available automation is either general purpose tools (e.g., word processors and graphic drawing tools) or special purpose tools supporting more formalized methods (e.g., simulation or CASE tools). Many semi-formal methods that may be custom to a particular organization.

Specific requirements engineering needs include the following:

- New techniques are needed for dealing with the volume and complexity of mission, systems and subsystem requirements.
- Automated support is needed for custom, semi-formal requirements modeling techniques.
- New approaches are needed to incrementally capture, elaborate and formalize requirements.
- The requirements engineering process must be "re-engineered" to reflect it as a continuing, iterative process, rather than a discrete step in system development.
- The effort to identify, understand, propagate and respond to a requirements change must be significantly reduced.

System Design & Allocation. System design has many of the same needs of requirements activities because design is an extension of a large and complex system definition problem. System design defines the requirements (allocated plus derived) for the first tier segments or subsystems. Subsystem design defines the requirements for lower level subsystems or components.

The system design process is also very dynamic, involving significant iteration and change. Moreover, the design process introduces a multitude of alternatives and tradeoffs that must be considered, weighed and analyzed. In many cases, important alternatives or tradeoffs are not considered due to cost and schedule pressures. This leads to stories of design by "the seat of the pants."

Negotiation and compromise are common during the allocation process and continue, to some extent, throughout the life cycle. Change impact analysis and management becomes critical for large systems.

Specific system design and allocation needs are as follows:

- Support is needed to effectively define, evolve and view a system design from many perspectives and from the aspect of the various subsystems and specialty disciplines.
- Automated support is needed for custom, semi-formal system design techniques.
- Requirements allocation must be made more efficient and traceability must be maintained throughout the development.
- Once allocated requirements have been baselined, anticipating, controlling, tracking, and managing changes to the baseline must handled reliably and efficiently.
- Automated techniques are needed to allocate requirements and design constraints or non-functional requirements to subsystems in a manner to guarantee that they will not be violated.

Interface Definition & Integration. The management and support of engineering activities associated with system and subsystem interfaces is a priority concern for the system engineer. Key interface problems exist in the following:

- Identification and understanding of interface requirements
- External interface adversity (i.e., complexity of interface interactions with the surrounding environment) [DEU90]
- Proper, complete and consistent interface specification
- Early negotiation of interfaces and proper management of the interface definition throughout the life cycle
- Change propagation from interfaces and across interfaces

Specific interface definition and integration needs are as follows:

- Interface requirements and definitions need to be controlled from the earliest identification and partitioning of the system.
- Automated support is needed for better managing the many complex interrelationships that exist across interface boundaries.
- Support is needed for managing and negotiating interfaces amongst many participants.
- Automated change impact analysis is needed to assist in assessing the impacts of interface changes and propagating the affects to the appropriate subsystems.

- Better specification and design representations are needed for complete interface specifications.
- Automated techniques are needed for guaranteeing the consistency of subsystem specifications and designs to baseline interface definitions.

Tradeoff Analysis. The major problem with tradeoff analyses is the cost, time and effort to accomplish them. A tradeoff analysis can represent a major diversion in the process simply to "answer a single question." Poor decisions are often made because of the lack of proper analysis or because sufficient alternatives were not thoroughly analyzed.

Typically, once a tradeoff analysis is accomplished, only the resulting decision is remembered. If the issue needs to revisited, the previous analysis is not available and the rationale for the decision is lost. As a result, even good analysis may be rendered useless if later factors change the decision arbitrarily.

Specific tradeoff analysis needs are as follows:

- Tradeoffs and decision rationale need to be captured and accessible for later review or reconsideration.
- The tradeoff process needs to be more productive in order to stimulate more thorough analyses.
- Automated support for tradeoffs need to support multi-disciplinary group processes that occur in a concurrent engineering approach.
- Integration mechanisms are needed to better support the use of multiple analysis tools and the synthesis of their results.
- Traceability needs to maintained between tradeoff decisions and the areas that they impact.

Engineering Decision Making. There are countless decisions made throughout a system development at all levels. Today, the decision making process primarily relies on human actions and interactions that often fail or insufficient in large organizations. Specific needs in this area include the following:

- Each decision needs to be specifically identified with all of the areas that the decision impacts.
- There is a need to reliably disseminate the decision to all those affected to direct a correct, consistent and unified response to the decision.
- The interrelationships between decisions need to be better understood.

- The resulting changes and actions from a decision need to be tracked to insure that the decision is completely and consistently reflected throughout the system development.
- Recording the decision and supporting rationale for decisions is needed to enable the inevitable revisiting or retraction of decisions.
- Utilities are needed to search and browse the decision history of a project.

Change Impact Analysis & Management. Effective change impact analysis is area of great need that would have significant benefit across the life cycle. Currently, simple traceability matrices are the systems engineer's only impact analysis tools. Effective change impact analysis goes far beyond simple traceability. Needs in this area include the following:

- Traceability needs be automatically captured and maintained as part of the specification, design and allocation processes.
- There is a need for more sophisticated and thorough means for identifying exactly what is impacted as a result of a proposed change.
- More reliable mechanisms are required for assessing, in terms of cost, resources and schedule, the program impact of changes.
- Support is needed for managing the change process across many individuals, in different organizations, and across various subsystems and work products.

Integration Planning and Management. Early in the life cycle, the systen engineer is involved in integration planning and in interface definition ana management. Integration builds must be planned consistent with individual subsystem development and supplier schedules. Interfaces must be completely and consistently negotiated and specified.

Later in the life cycle, the svstems engineer is faced with two significant challenges:

- Maintaining the consistency of the interface definitions in response to numerous changes
- Assessing the compliance of subsystems or suppliers in adhering to the interface specification and not violating any other total system design constraints

The systems engineer attempts to anticipate and respond to integration problems before they occur. Once integration has begun, the systems engineer is in a tracking and reactive mode, attempting to minimize the impacts of integration problems on the overall system cost, schedule, or mission.

Specific needs in this area include the following:

- Better mechanisms are needed to couple the integration planning activities with the design process.
- Automated support is needed to maintain the consistency of integration plans as the design changes.
- Integration impacts need to identified from other system baseline changes.
- Automated techniques are needed for guaranteeing the consistency of subsystem specifications and designs to baseline interface definitions, even if those design specifications are controlled by remote subcontractors or suppliers.
- Supplier management needs to be better integrated into the development process.
- Contingency management is needed to better insure that integration schedules are met.

Quality Engineering and Assurance. Quality assurance, imposed from the start and throughout system development efforts, is needed to reduce the risk of failure. [AFS89] There is a great need for the automated support of "total system quality" in systems engineering. Assurance is becoming an increasingly more difficult task for large and complex systems, while at the same time becoming more essential for safety, security, and trustworthiness.

Specific needs in this area include:

- Quality needs to be driven by and tailored to specific project needs.
- More efficient ways are needed to accomplish incremental review and critique of information.
- Automated metrics support is needed for the measuring key quality aspects of work products and the process.
- Better ways are needed to view, compare and analyze information or metrics data from different sources.
- Automated techniques are needed to focus human attention on only those problems areas or areas of interest.
- Statistical quality assurance techniques need to integrated throughout the development process.
- Automated support is required to alert those individuals necessary to take action of critical events, metrics or observations.

Specification Generation. Documentation often accounts for 20-30% of the cost of a large system development. Yet studies have shown that MIL-SPEC documentation is not an effective communication mechanism.[KRA88] Automated methods are needed to significantly reduce the unnecessary cost of specification development and production. Specific needs in this area are as follows:

- Support is needed for better understanding and tailoring of data item descriptions and requirements.
- Better automated support is needed for describing and maintaining document templates and contents to automated documentation tools.
- Bi-directional automated support is needed for reflecting changes in edited documents and the sources for the information.
- Better forms and representations are needed for communicating information for review and analysis.

2.2.2 Communication Needs

This section discusses the communication needs for the following activities:

- Collaboration & Coordination
- Information Research
- Boundary Spanning
- Joint Work Product Development

Collaboration & Coordination. The system development involves a large number of groups to coordinate their activities, or at least share information, during development. Early phases concentrated on clarifying issues, defining terms, coordinating representational conventions and creating channels for the flow of information. Several impediments to communication were discovered. [KRA88]

- The complexity of the customer interface hindered the establishment of stable requirements
- Organizational boundaries hindered understanding the requirements
- Political barriers created a need for informal communication networks
- Temporal boundaries buried design rationale

The four types of common communication breakdowns that were found are: [KRA87]

- No communication between groups

- Miscommunication between groups
- Conflicting information from multiple sources
- Communication problems due to project dynamics, including:
  - information loss during transitions between phases
  - loss of unrecorded knowledge due to change or loss of personnel
  - exponential growth of potential communication paths as the project becomes large

Specific collaboration and communication needs are as follows:

- Broader automated support is needed to support multi-media, group communications.
- Information capture and recall facilities need to be better integrated into electronic communication mechanisms.
- Communication utilities needs to better support formal communication process requirer.:ents.
- The information sharing process needs to include the automatic notification of selected individuals of new, important information.
- Utilities are required to better organize, prioritize, manage volumes of electronic mail.

Information Research. The systems engineer utilizes a broad spectrum of information: reference information, information particular to a program, and information about previous programs. The information research needs can be characterized as having three parts:

- Capturing or providing electronic access to a wide variety of heterogeneous, multi-media information
- Providing facilities for effective search of specific information from the resulting vast store of information
- Easy retrieval and application or reuse of information assets

Information research needs include the following:

- The vast amount of information that is generated during a system development needs to be automatically captured, indexed, and organized.
- Application and system knowledge needs to be readily available to the various members of the system development team.

- Meaningful classification information (e.g., keywords) needs to be automatically extracted from reference information.
- New user interface paradigms are needed to promote searching and browsing large, complex heterogeneous information stores.
- Automated assistance is required by users in formulating, executing and refining search strategies.
- Better approaches are needed to extract, adapt and reuse information fragments.

Boundary Spanning. Current automation, at best, supports information sharing. New approaches are needed for supporting and facilitating group interactions electronically and supporting group processes. Specific needs include the following:

- Practical electronic substitutes for meetings are needed to lessen the burden of group coordination.
- Support is needed to reconcile different jargon and terminologies that occur when multiple disciplinary interaction occurs.
- New user interface paradigms are needed for collaborative group processes such as brainstorming and negotiation.
- Standards are needed to support "groupware" across heterogeneous computing platforms.
- Innovative ways for detecting and responding to communication and coordination breakdowns are needed.

Joint Work Product Development. Current document product automation does not explicitly recognize that many work products are group produced. Automation needs to specifically address group work product development, as follows:

- Support is needed for convenient work product partitioning, allocation and assignment.
- Information editing facilities need to support controlled simultaneous editing of work products.
- Automation is required for the work product review, markup and markup synthesis process.
- Support is needed for coordination and tracking of work product contributions.
- Automated support is required to synthesize work products from individual contributions.

2.2.3 Management Needs

This section discusses the management needs for the following activities:

- Standard and Policy Application
- Process Management
- Program Planning & Tracking
- Task Management
- Risk Analysis

Standard and Policy Application. In practice, many contractual guidelines and/or details are not carried out, or adhered to. These guidelines are generally specified in great detail for good reasons, yet very often are not implemented (i.e., lost in the shuffle), to the detriment of the project. Deciding what standards and DIDs to include on a contract is a difficult and confusing task. Needs in this area include the following:

- The various standards, DIDs, policies and guidelines that apply to project need to be available on-line for query and browsing.
- Support is needed for process modeling and tailoring that is compliant with the various standards and policies.
- Expert assistance is needed in the standards tailoring and synthesis process.
- Automated means are needed to guarantee continued compliance with important standard and policy provisions throughout the system development.

Process Management. There is a need for innovative acquisition *tailoring* and alternative development processes such as incremental development and prototyping. For example, "user involvement should be tailored for each program, varying from cases requiring very limited involvement to ones in which a user will assume the lead role." From user involvement to development process models, it is acknowledged that no single acquisition strategy can possibly serve all situations. [AFS89] Process management needs include the following:

- Automated support is needed for process creation, assignment, execution, enforcement, and tracking.
- Support is needed for preparing standard or prescribed process templates and their application and tailoring.

- A variety of process guidance and enforcement approaches are needed to deal with different situations.
- Process automation needs to support informal, as well as formal, processes.
- Process management needs be be integrated with program management automation.

Program Planning & Tracking. Systems engineers will inevitably do some level of planning, scheduling, estimating and tracking, although the amount of planning and tracking activities will vary by individual and by organization.

Key problems in project planning and tracking include:

- Accurate and reliable estimation
- Scheduling within constraints
- Staff planning
- Interface to various reporting systems
- Managing plan changes

Key needs in this area include the following:

- Program management automation needs to be more flexible to support plan manipulation in any of the various plan representations.
- Support is needed to reconcile plans at various levels and to maintain consistency between high level and low level plans.
- Program management needs to better support contingency planning and management.
- Making changes to plans and executing those changes needs to be streamlined.
- Automated support is needed to assist in generating and evaluating alternative plans within defined constraints.
- Support is needed to detect interrelationships and conflicts in separate organizational and program plans.
- Automated support is needed to translate between different institutionalized program management and accounting systems.

Task Management. Because of the multi-disciplinary nature of the concurrent engineering process, careful tasking and monitoring of the activities of a may individuals must be accomplished. Key problems in tasking include:

- Conveying and enforcing the proper process for the task
- Maintaining proper priorities
- Effective time management
- Accomplishing optimal staffing and tasking of the staff
- Monitoring and tracking progress

Task management needs include the following:

- Support is needed to quickly define and assign new tasks in a group collaboration environment.
- Task monitoring and notification support is required.
- Individuals need automated aids for task multiplexing and dynamic prioritization.
- Group scheduling support is required, particularly in identifying available meeting times.
- Dynamic task delegation and partitioning is required.

Risk Analysis. The proper management of risks is critical to the success of systems engineering. Unprecedented systems—"systems for which there has been no precedent in the form of similar systems or systems performing the same functions, or for which the design teams lack full or applicable system design experience"—should receive special attention with regards to risk reduction. [AFS89]

Efforts should be made to identify and control risks as early as possible in systems development. For example, policies should mandate the use of risk management plans. Proper risk-driven analysis of alternatives may at any time redirect the development effort in terms of rework, step sub-setting, or specifying/adjusting design-to-cost levels-of-effort.

Specific needs in this area include:

- Automated support for risk management is needed that includes the identification, analysis and tracking of potential risks.
- Risk analysis needs to be integrated with contingency planning.
- Metrics and monitors needs to be supported to automatically detect risky conditions.
- Alternative plans and contingencies need to be quickly retrieved and executed when required.

3. Summary of Process Model

FTR, Volume 3 - Process Model presents the details of the results of the work performed by SECD subcontractor, Alberto Ortiz, McDonnell Douglas - Douglas Aircraft Company, Long Beach, CA. The following areas were examined in order to better understand the automation requirements:

- Process modeling work of other corporations
- Process modeling methods and tools
- Systems engineering taxonomy
- System engineering standards
- Process modeling field interviews
- Process model trade-off study
- Summary of technical library searches
- System engineering researched documents database

The culmination of the Process Model effort was a SECD Generic Process Model. This process model was implemented and demonstrable on a Apple Macintosh IIfx Workstation using MacFlow, a COTS product. The process model is important for three reasons:

1. Provides a better understanding of the breadth and cope of system engineering activities in a military system life cycle
2. Used to verify the completeness of Catalyst system requirements
3. Can be tailored or adapted to meet a specific organization's needs

The SECD Process Model Task was arduous and demanding in all facets. This interactive system life cycle Process Model is extremely comprehensive, flexible, and extensive. It is intended to support the necessary degrees of freedom needed to accommodate the many identified life cycles standards, as well as, new and emerging ones. Its strength lays in the identified interfaces between government acquisition and contractor development and the interaction between the system engineering process and the system life cycle activities. The following are our conclusions about the Process Model, system engineering, and the system engineering process:

- The system engineering process is contained within the system development process, which in turn, is contained within the system acquisition process.

- The system engineering process cannot be separated from the system development process. The interfaces between system acquisition and system development must identified for the system engineering process to be meaningful.

- System Engineering includes Concurrent Engineering (CE), Reverse-Engineering, Re-Engineering, Requirements Engineering, Integrated Process Development (IPD), Design To Life Cycle Costs (DTLCC), Technical Program Management, System Architecture, System Analysis, and Commercial Systems. System engineering does not include hardware or software development or system integration. These disciplines are considered specialities within the Process Model and they have a big impact in the system engineering process and its work products.

- The system engineering process varies from one organization to another. Its variation encompasses the entire system life cycle horizontally and vertically. Variation of the system engineering process is a key issue in the Process Model Task.

- System engineering processes and methods are derived from accepted industry standards and include best practices for implementation. The Process Model is the means by which the users are empowered to tailor and improve system processes and methods.

- The Process Model, by virtue of its extensibility and completeness, can be used to customize systems engineering processes unique to each individual development approach.

- The Process Model can be used as a Reference Model to map system engineering methods and assess their completeness and effectiveness in the development of a system.

4. Summary of Interface Standards Studies

Software interface standards offer benefits to software producers and users despite being sometimes conflicting and time consuming to develop within standards organizations. When properly coordinated and publicized, they convey information, optimize variety, improve quality and promote compatibility, interoperability, and transportability. Improvement of documentation, interfaces, interchanges, processes, and procedures can be the net result of using interface standards. Efficiency and cost reduction can also be benefits, if standardization does not occur too early or too late. Premature standardization can increase costs in the long run by inhibiting innovation and competition. Standardization too late results in proliferation causing software production costs switching to the new standard [NAS86].

An interesting side effect of standards is the acceleration of the arrival of cheaper computer technology. When a standard begins to appear in draft proposals, the software developers can address the problems of a particular system and build solutions to those problems. The hardware engineers then become a part of the development team effort to produce the same problem solution in firmware. Once the issues are identified and defined, the industry can begin the process of putting the standards into silicon, and therefore, accelerating the reduction of prices in computer technology [MAG90].

The key to the success of Catalyst is in the interfaces within and between environments. Therefore, the recommendations and eventual choice of interfaces used in the SECD environment was critical. Conformance to national, international, DoD, and industry de facto standards (e.g., CALS, PDES, POSIX, GOSIP, PHIGS, DoD standards, X-Windows, OSF/Motif, PostScript, CEF, CDIF, EDIF) will aid in accomplishing integration of the Catalyst Environment and result in high paybacks. The risk lies in that some areas of standards are insufficient in 1992, but are continuing to develop.

FTR, Volume 4 - Interface Standards Studies presents the evaluation of the standards for each of the three SECD defined interfaces (i.e., framework, tool, and virtual machine). Inc!uded below are the conclusions and areas for future research and development in interface standards.

4.1 The Frameworks Interface

4.1.1 User Interfaces

A review of User Interface technologies in current literature indicated that standard workstation fare was mouse-driven, multi-windowed, using text, graphics, and icons. The user interface that is recommended for the Catalyst Environment is the X-Window System, OSF/Motif, and PHIGS. Extensions to PHIGS, known as PHIGS+, defining the surface rendering extensions to PHIGS (ANSI/ISO), and PEX supporting PHIGS implemented under X, the PHIGS library and X Windows are all recommended for the Catalyst Environment.

To address the other side of the user interface as described in the user interface model, User Interface System <-> System Engineering Catalyst System, we evaluated class libraries. Class libraries are an emerging technology and may be of limited use in five-year time frame. Current software libraries are limited to low-level utilities to support standard interfaces. A large base of desired software components is not yet available. Object-oriented languages were presently the most the mature with some having commercial class libraries.

In the months and years to come, we expect to see increased development of user interface technology and component libraries designed for reuse. Areas for growth are extremely high-resolution images, multimedia application, full-motion video, and new ways of interacting with data. Intelligent interfaces may not only help the user to automate everyday tasks, but may even anticipate the user's actions and thereby increase productivity [HAY89].

4.1.2 Communications

Much work has been done in standardizing a model for communication interfaces and has made network technology mature and rapidly proliferating. There are many commercially available products, both in hardware and in software. Of the three classes of interfaces in the Catalyst Environment, the standards of the Communication Interface were the most mature.

Government OSI Profile - OSI Networking Layers GOSIP (FIPS 146) is recommended for the Catalyst Environment as the communications interface. Established in August 1988 and mandated for Federal Government agencies in 1990, the GOSIP standard is compliant with OSI and provided the stability of a standard for the future implementation of a systems engineering tool. Data communications in the CALS effort has identified GOSIP as a model for the OSI environment to facilitate truly integrated information systems.

4.1.3 Repositories

Even though SQL, ANSI X3.138, and ISO IRDS are established standards for relational databases, we do not feel comfortable recommending these standards that use the entity-relationship paradigm or relational tables to produce a relational database for the Catalyst Environment. A review of database models indicates that they support functional technology with mature relational approaches, E-R being the most popular. Instead, we recommend using an object-oriented database for Catalyst; however, currently there are no firm established standards for object-oriented databases, but the future trend is towards that goal. PCTE+ and ATIS both extend the entity-relationship approach with object-oriented concepts and are strengthen by the added generalization and specialization. The OMG's object broker and emerging standard appears as the best choice for demonstration and validation of Catalyst, a state-of-the-art system engineering environment.

With the rapid evolution of the technologies for tools, methodologies and repositories, the prospects for a single, comprehensive official repository standard is probably at least 3-4 years in the future [FOR89]. The object-oriented databases (OODB) are better suited to CAD/CASE/CAE than relational approaches. OODBs rely on fundamental infrastructure and integration technology. The appeal of OODBs are that the data model more closely matches real-world entities, and the database language can be integrated with an object-oriented programming language. The object-oriented database technology has an emerging set of commercial vendors with many commercial products. However, large-scale applicability to multiple CAD/CASE/CAE products presents some risk at this time.

In the meantime, competitors in the CASE industry will incorporate repository technology that enables them to deliver advanced functionality, and in many cases, that will be proprietary technology. The evaluation of the repository products indicates no certain winner for a particular product or vendor, each has their own interesting features. Versant and Objectivity, both supported by the OMG, are promising choices for object-oriented databases.

However, the underlying models for all the proposed repositories are very similar and can be seen as a positive result of the early attempts at standardization meaning that the differences are relatively manageable. Consequently, it may be possible to build bridges between the major official and de facto repository standards such that repository information can be transferred among systems as needed. Of course, this will not provide the fully-open, plug compatible CASE environment across all tools, vendors and platforms that CASE users would like to see. However, it will protect and preserve the valuable information assets developed as organizations expand their use of CASE tools and methods.

4.2 The Tools Interface

We recommend that Catalyst support an ASCII object-oriented data interchange; CEF and clipboards for tool interoperability; link databases and Object Request Brokers bear close watching as they develop; PDES for data exchange, CDIF for data interchange formats, EDIF for design interchange formats, and PHIGS. Extensions to PHIGS, known as PHIGS+, defining the surface rendering extensions to PHIGS (ANSI/ISO), and PEX supporting PHIGS implemented under X, the PHIGS library are all recommended for the Catalyst Environment. Postscript is recommended for printed and displayed page description, and SGML for document representation. All tool interfaces and standards support CALS. We recommend that Catalyst support translation of its object-oriented data interchange format to/from dominant Government and industry standards, as they become more well-developed, for data interchange of engineering information.

Because many types of graphic storage formats exist, it is not clear whether a useful universal design-graphics representation can ever arise. A standard tool interface must accommodate many formats:

- Bit-mapped formats (e.g., PCX and TIFF files, which may be created by scanning photographs, or converted from other formats)
- Line-segment formats (used in engineering drawing programs such as AutoCad and others, principally for ease of output to plotters)
- General object formats (as used in freehand drawing programs such as Micrographix Designer, Corel Draw and others)
- Proprietary object formats (for special design functions such as integrated circuit design)
- Page-description languages which can incorporate both text and bit-mapped or object graphics (HP/PCL Postscript, and the proposed Tru-Type are examples)

A key issue with graphics is the degree to which, and how, the graphical format can be tied to meaningful parts of the object(s) represented. Some graphics formats use layers to permit the graphic to be separated into different parts representing different parts of aspects of the system. Probably the most satisfactory engineering graphics are those which represent two-dimensional objects such as the photoresist layers for creating a semiconductor integrated circuit. Graphics which represent three-dimensional objects are very far from standardization; for purposes such as envisioning the view from a particular point in a building, these graphics can be most helpful [BEA90].

The bottom line is that, at this point in time, attempting to standardize graphic formats to a small set is not yet feasible. Even if the system engineering environment were to be fully devoted to C-cubed systems and nothing else, it may not be possible to standardize graphics meaningfully. This has a major impact on the tool to tool interface. This implies that the systems engineering environment needs a very capable conversion scheme which will permit viewing graphics in various formats when embedded within text, and the ability to separately enlarge graphics to a separate window or screen for easier viewing or manipulation while the associated text is studied.

4.3 The Virtual Machine Interface

UNIX is becoming a major driving force in the area of workstation/environment frameworks. In many cases, the environment framework is built around UNIX (e.g., SUN NSE, Apollo DSEE) and in other cases it is built on top of UNIX (e.g., PCTE, GENOS). Because UNIX is characterized as a stable, de facto standard, the UNIX operating system is recommended as the virtual machine interface.

To support the choice of UNIX as the operating system for Catalyst, we recommend adhering to the POSIX standard. This standard defines a standard operating system interface and environment to support application portability at the source code level. The trend towards convergence of different versions of UNIX will accelerate the evolution of the POSIX standard. Sun Microsystems and AT&T have a joint effort to converge the Berkeley 4.2 and AT&T System V Interface Definition implementation of UNIX. A similar effort between Microsoft and AT&T to converge Xenix and System V is underway. Other major framework technology areas include windowing interfaces, database interfaces, network and data communication interfaces.

4.4 Future R&D of Interface Standards

A major reason standards take so long in development is one that also delays many product deliveries — creeping functionality [CHI88]. If new procedures for standards were currently adopted to ensure that development groups are well managed, standards will no longer be delayed by the continuing cry for "just this last function." Emphasis on a reference model, requirements work, and new work item justification should focus the efforts of experts working on standards development. This emphasis may also provide a clear scope and goals statement for the work to be done. No longer should a standard be defined to attain more than one goal. In the past, some standards have been developed with diverging goals. All these new procedures could lead to better production of standards, and hopefully be applied to future standardization processes.

Although standard architectural interfaces are highly desirable, these standards are still quite elusive. A number of competing efforts make it unclear as to what direction standards are really taking. For example, at the CASE 88 workshop, one hundred twenty-seven people met to discuss and recommend different standards. It is clear from this effort that industry wide standards are still a long way off. The following is a list of some of the different standards efforts [RUD89]:

- IEEE Task Force on Professional Tools
- IEEE P1003 Portable Operating System Interface for Computer Environment (POSIX) effort has been going on for several years. POSIX has been a trial-use standard for almost two years and has received an affirmative ballot as a full-use standard. A final ballot is expected shortly and is expected to be approved.
- ANSI X3H4 committee on information-resource directory systems
- U.S. DoD Common APSE Interface Set
- ESPRIT PCTE environment standard
- European Computer Manufacturers Association
- CASE technology subcommittee of the Electronic Industries Association's EDIF standard
- ISO committee SC7 (software development and system documentation)
- Digital/Atherton tool-integration services proposal
- National Bureau of Standards
- Software Productivity Consortium
- Object Management Group

All these individual efforts, as well as others, will continue to define interfaces in a variety of system areas and will affect engineering decisions concerning the implementation of Catalyst. The following is a discussion of future trends for R&D of interface standards.

Window Interfaces: X-Windows, based on MIT's X11 system, is gaining wide acceptance in the scientific, engineering, and commercial communities. It has been adopted by major computer vendors like Digital Equipment Corp., Hewlett-Packard, Apollo, Masscomp, and Tektronix.

Network and Data Communication Interfaces: Open network interfaces are critical for allowing networks of heterogeneous computers to communicate effectively. This field has a number of competing approaches, including Ethernet, TCP/IP, OSI, SNA. Major workstation vendors like Apollo and SUN have their own open network architectures (NCA and NFS respectively).

Data communication is critical to geographically distributed development. More advanced protocols are needed as we migrate from file-based to database based frameworks. There is a clear trend within the international community towards the OSI standard. Major computer vendors are aligning their network protocols to OSI. The primary exception is IBM with a large SNA installed base and the

U.S government that is using TCP/IP for all their communications needs. The adoption of a universal standard communication protocol is a few years away.

Recommendations from the IDA for interface standards are that they are difficult at application level as well as the semantic and syntax at the Service Interface Level. Interfaces need to be defined at all levels, and consequently, need more study.

Database Interfaces: In the database area, interface standards are still under development. Some CASE vendors have adopted relational databases to gain wider customer acceptance, even though relational technology is not a good fit for engineering applications. CAIS and PCTE have variants of the entity-relationship data model. More powerful object-oriented database products are emerging and will become suitable platforms for CASE products and environment project databases. Object-oriented databases offer an increase in expressive power which is more suited for complex engineering data.

One database standard that may have an impact on environment frameworks is the Information Resource Dictionary System (IRDS) developed by the ANSI X3H4 committee. IRDS provides a standard, data model-independent means for describing data. Rather than focus on a standard data model supported by standard database interfaces, IRDS offers more flexibility in choosing an underlying database by standardizing the data dictionary. This standardization allows data definitions to be shared among tools. In this respect, the efforts to standardize data models and data management interface, as proposed in the CAIS and PCTE interface standards, is of limited value and prone to obsolescence with the rapid advances database technology. IRDS may provide a framework independent solution for data integration.

If there's a black hole in the CASE universe, it's the repository. Perhaps because of its central role in the CASE integration architecture, the repository tends to touch almost every aspect of the system engineering environment. Consequently, any discussion that starts with the repository can lead in just about any (or every) direction, and end in a parallel universe.

Expect OODBs to penetrate certain database-application markets for which RDBMSs have proved unsuitable. Object-oriented databases are more than a passing fancy. Also, expect the object-oriented approach to make traditional programming techniques obsolete. System integrators must think about objects. As repositories become dynamic in nature, they will take over many of the tasks of operations management, providing system configuration information at run time to eliminate most of the need for job control language. Late binding of applications to information in the repository will ensure that all applications are up-to-date and will provide a degree of flexibility in applications that has not previously been possible with applications bound at compile time.

Repositories will also become the site for advanced prototyping facilities, such as the ability for business managers to simulate proposed business processes and policies and observe the results before requesting applications from the develop organization. Repositories will also manage the output of reverse engineering tools so that developers can easily examine and re-engineer existing applications with a variety of maintenance tools. Repositories will also become the place where reusable components, in the form of both designs and code, are made available to developers for future projects. The proximity of requirements, proposed designs and existing designs increases the probability that developers will look to reusable components as a viable solution to the software challenge.

In the engineering environment the trend toward consolidating all aspects of product development and manufacturing will continue as software development is integrated with microcircuit design, printed circuit board design, mechanical engineering and manufacturing databases. The opportunity to reconcile software development with total quality management initiatives in other disciplines hinges on the ability to collect accurate statistical control information, a requirement that should be greatly facilitated by the CASE repository. Other benefactors of the repository database include field support, R&D and technical publications departments.

However, OODBs do face barriers to acceptance. First, they're up against a large installed base of business RDBMSs, while commercial OODBs are only just starting to appear. Second, object-oriented standards have not yet jelled, although several groups are working on defining object-based programming language, OODB terminology and interface standards.

The market of OODBs is still small. However, this market should grow rapidly because OODBs give companies the capability to manage certain types of information, such as text, graphics, voice, and video, that relational databases are not geared to handle as well.

In the future, the range of information stored in the repository will expand to include enterprise information such as business data models, business rules and processes, strategic business planning, test management, and software quality assurance. The repository will contain the definition of enterprise information architecture, eliminating redundancies and ensuring that inconsistencies are resolved. Intelligent database technology is an specialized area of database research and deserves future investigation.

5. Summary of Technology Assessments

Conclusions from the Technology Assessments appear here in FTR, Volume 1- Effort Summary and also in FTR, Volume 5 - Technology Assessments. Volume 5 provides the detailed assessments of technologies related to peripherals, workspace software, framework software, and tools.

5.1. Storage devices

Magnetic media in general is a very stable technology, and is not likely to be entirely eclipsed by other media, such as optical. Disk drives should continue the current trend toward greater capacity and throughput. The smaller form factors will become more popular, partly due to the spread of notebook-type computers. Tape will likely continue to be the favorite form of back-up and archival media. The newer, smaller form factors—8mm and DAT—will likely gain more adherents.

Optical media should continue to grow in popularity. CD-ROM is relatively stable, and has been since the introduction of the High Sierra format standard. More and more software and/or data is being distributed this way, and some vendors—notably, Apple—discount the prices of CD versions of their products. There could be improvements in access time and throughput as R&D contributes to lighter, less bulky mechanical components in the read arm.

WORM drives, with their lack of rewrite capability, may end up a niche technology, popular primarily with those industries that need unalterable audit trails; examples are the financial, insurance, and medical professions. The lack of format standards will continue to prevent interchange of cartridges between drives from different manufacturers. This may also prevent some companies from buying; they won't want to become locked into a single manufacturer's product and pricing/support strategy. As with CD-ROM, R&D in smaller components may improve the access time and throughput.

EO technology is still in flux. Magneto-optical drives drives are available now, and format standards are in place. However, R&D into alternate methods, such as phase-change and dye-polymer media, could put competing technologies on the market. Standards may permit cartridges to be used on the drives of different manufacturers, but they certainly will not permit magneto-optical cartridges to be read in drives that use dye-polymer cartridges.

5.2. Input devices

Although there is a great deal of R&D in alternate input technologies, the keyboard will be the mainstay for a number of years to come. The mouse—and its alter-ego, the trackball—will continue to be the most popular pointing device, although some users may move to graphics tablets. That move depends on the price of the tablets, currently much higher than the cost of a mouse, and whether the users have other uses for the tablet. Graphics designers and CAD operators will probably continue to be the largest consumers of graphics tablets. All of these technologies are stable, although it is possible that R&D could create mouses and graphics tablets with resolutions higher than those currently available. The pen-based computer technology is currently testing the market to see if users are ripe for a new input dev ice.

Voice input is getting plenty of press, and thus plenty of attention from the marketplace. As prices come down and capabilities increase, these devices may come into more common use. For now, they are a significant first-cut technology, with a great deal of R&D still going on. Expect to see great strides in audio and voice technology in the next 5-7 years.

Scanners are growing in popularity, especially since more capabilities are appearing at lower prices. The primary users will be those who need to capture images and include them in their data/documents, and those who will benefit from OCR applications. Scanner technology is stable, and OCR is relatively stable. However, R&D in neural nets and other areas of artificial intelligence could bring great improvement in OCR's recognition capabilities. At present, the error rate keeps OCR from being any faster than typing.

5.3. Output devices

As a whole, the technology of the current generation of output devices is stable. However, R&D can bring many improvements to them all. It remains to be seen whether the improvements will make the current devices obsolete.

We could see improvements in display quality at all price levels; certainly the displays on most users' desktops would have once cost as much as or more than an entire computer system; color is available at what monochrome once cost With the growth of imaging applications, the market will be looking for higher resolution. Larger displays are likely to become more common in the office market, as they have in the workstation market.

Laser printers have not completely replaced other printer technologies, but they have become the "standard" against which others are compared. Coming improvements include higher resolution and less expensive color. Prices are already dropping, and may continue to do so. Larger companies with a large volume of printing will be watching the developments with Canon's color laser printer/copier.

Imagesetters and film recorders will probably remain the province of high-volume users, with others willing to make use of a service bureau. However, that could change if costs and ease-of-use should drop down to the high end of laser printers.

Plotters are likely to remain primarily CAD/CAE tools; their throughput is a disadvantage for other types of graphics producers, who also demand more versatility in their output devices. The newer technologies may drop somewhat in cost, but budget-conscious, smaller operations will keep the market for pen-based and electrostatic plotters alive and well.

The technology for getting digital images onto videotape is stable, but remains fairly expensive. As desktop video takes off, the demand for equipment could drive prices down, as is already happening with some of the video add-on boards. As always, though, the more sophisticated features you want, the more you have to pay.

5.4. Operating Systems

There are conflicting opinions about the future of operating systems. UNIX has been around for years, and many pundits expect it will stay on for many more. However, there is much research into alternatives, some based on UNIX but offering desired capabilities, such as greater security. Distributed operating systems will likely become more commonly used, due to the growth of distributed systems. UNIX will not disappear, but it will be an option, rather than the only choice.

If parallel computing becomes more than a niche, parallel operating systems may take up a larger share of the market.

5.5. Repositories

The relational database seems to be the most common in use, according to the products on the market. Object-oriented databases may be the coming thing, given the move toward object-oriented programming and modeling techniques; both need storage and management for objects that do not fit well into relational structures. Relational is not likely to disappear soon, and in fact may be used in conjunction with object-oriented forms of storage. The work being done in federated databases is an indicator of this.

5.6. User Interfaces

Applications are becoming more complicated, while there are more and more users, both sophisticated and otherwise. The sophisticated users demand easier-to-use interfaces so as to increase their productivity, and to make complex operations easier to control. Less sophisticated users also want easier-to-use interfaces, so as to make computers less intimidating and to shorten their learning curve, as well as make complex operations easier to understand. The graphical interface seems to be here to stay. The questions still being worked on are ones of detail rather than of concept. There will probably continue to be several "standards" for graphical interfaces, but object-oriented tools developers will be able to create a single interface and quickly port it to any desired platform and GUI. The only real concern is whether every GUI vendor will wind up in court, paying legal fees, fines, and royalties to Apple for using icons and movable windows, thus raising costs to the end user.

5.7. Hypertext/Hypermedia

The current generation of hypertext/hypermedia tools is fairly stable, and offers many usable capabilities. R&D continues, however. The first result will likely be better interfaces among different media sources, and tools that are easier-to-use. The main problem will continue to be design and implementation. Hyper-whatever is not the panacea that some people want it to be or that some vendors claim it is. For some problem domains, and with good up-front design, hyper-technology will solve problems and make systems easier to manipulate. For others, it will be the wrong choice.

5.8. Access Control/DIS Security

A great deal of R&D has been and is being done in this area. That will continue as long as one group wants to protect information and another wants to get at it. The complexity of modern distributed systems, especially those that are geographically distributed overs hundreds and thousands of miles, make it a difficult problem to solve. There are those who believe the only guarantee of security is to stay off the network. The current generation of commonly-used operating systems appear to be only as secure as their users; not much is done to enforce security measures. File servers such as Andrew could add some enforced security to an existing operating system such as UNIX. Operating systems that are under development seem to have security in mind from the start, and offer enforced security features as a matter of course. This fact may be a major reason for some companies choosing an OS other than vanilla UNIX.

5.9. CAD/CAE

As design tools, CAD/CAE packages are stable. More sophisticated features, such as solid modeling or interfaces to spreadsheets for estimating and billing, continue to appear. Features once found on more expensive workstation tools continue to migrate down to the PC tools, and the price/feature ratio continues to drop. Research in virtual reality has provided some benefit to this market, as well, with some packages now offering "walk-through" and load simulation capabilities.

CAD/CAE will benefit by greater connectivity between different types of software tools—such as the afore-mentioned interface to spreadsheets, for example. As various committees come up with standards for interfaces and data format, we may see a single system handling design, simulation and testing, analysis, estimating and billing.

5.10. CASE

R&D in CASE tools continues. Most such tools are used for analysis, for design, or to develop requirements. Unfortunately, most support a single method, while most organizations use several for different purposes. Another problem is that many tools provide no metrics or cost-estimating support. Rather than develop an all-purpose tool, however, the industry is more likely to continue to support efforts such as this, allowing them to use any tool in their arsenal from a common point, with access to all the design data from any tool.

The CASE industry has accomplished a great deal during its first decade, but there is still much more to be done. In, fact, the more software engineers and systems professionals become dependent on CASE, the greater demands on the technology.

5.11. Configuration Management

Configuration management tools are somewhat in flux right now. Systems are becoming more complex, and there is more to CM than just keeping a copy of the latest file. It is difficult enough to track thousands of software components through all their versions for a single platform; multiply the problem times X platforms and Y supported versions on each of them. CM tools need visibility into all the sources of software objects, and through-out the life cycle. Again, rather than build a single all-encompassing CM megalith, it seems more likely that "environments" such as this one will incorporate several different types of CM tools that will share a central data repository.

5.12. Groupware

Groupware, as software for computer-supported cooperative work is known, is the latest buzz-word. For the most part, it is still an R&D field, despite the availability of commercial products. In some people's minds, groupware means that you have multi-media access to any- and everybody and their data from your workstation. In reality, the commercial products are more limited in scope. R&D systems offer more of those fantasy features, but given the communication bandwidth needed for combined audio/video/data transmission, such systems are not likely to see common use in the near future. However, collaborative writing tools and multiple-person editors, group conferencing and decision-making systems, and the like should continue to penetrate the market.

6. Summary of Trade Studies

Conclusions from the Trade Studies appear here in FTR, Volume 1- Effort Summary and also in FTR, Volume 6 - Trade Studies. Volume 6 provides the detailed study of the SECD environment concepts and rationale, technology demonstrations, prototype scenarios, and risk analysis.

6.1 Catalyst Building Block Concept

The systems engineering automation focused on the development of high payoff *building blocks* for a systems engineering environment, rather than attempting to specify a complete and comprehensive automation of the systems engineering process. These "building blocks" consist of highly adaptable and configurable software tools and environment frameworks, that, when integrated with an installed computer system (hardware plus system software) and other software in the users' environment (tools and frameworks), provide an automated environment for systems engineering.

A complete, monolithic systems engineering environment (i.e. all tools and all required frameworks) was considered infeasible for a number of reasons:

- The systems engineering process, as practiced across the numerous types of mission-critical systems, is too broad to practically automate in a comprehensive fashion.
- There is a high degree of variability in the systems engineering process across organizations that precludes a single environment solution.
- Each organization already has a number of tools, frameworks and computing hardware that it has experience with and will want to continue to use.
- A single monolithic environment is not considered commercially viable; organizations prefer to pick and choose their tools from multiple sources.

The building blocks approach sought to identify a set of widely applicable and highly adaptable tools and environment frameworks that can be integrated with other COTS (Commercial-Off-The-Shelf) tools and frameworks, and internally-developed tools and frameworks to form an organization's systems engineering environment. This concept is shown in Figure 6.1-1.

Following this concept, an organization's systems engineering environment resembles distributed, cooperating *islands of automation* that are extensively adaptable. It is our hypothesis that automated tools transition faster and more effectively if they are flexible enough to simply automate the users' existing process, rather than requiring the users to change the way they do business. As a result, the tools and building blocks that are specified are *ultra-flexible,* supporting adaptation along many dimensions, allowing organization-specific definition of:

- Process and life cycle
- Organization structure and roles
- Methods and techniques
- Information and representations
- Work flows and work products
- Policies and procedures

Moreover, the tools and building blocks must be portable to a number of computing platforms and support integration with the organization's installed base of COTS and internal tools.

Each tool and framework building block should strive for power and simplicity within a narrow scope, yet be highly interoperable with other tools and framework building blocks. An integrated set of highly effective, single purpose tools was better than a single, monolithic environment that attempts to solve all problems. The integration mechanisms between these tools, whenever possible, was ubiquitous (using the framework building blocks), rather than being a recognizable layer to the user.

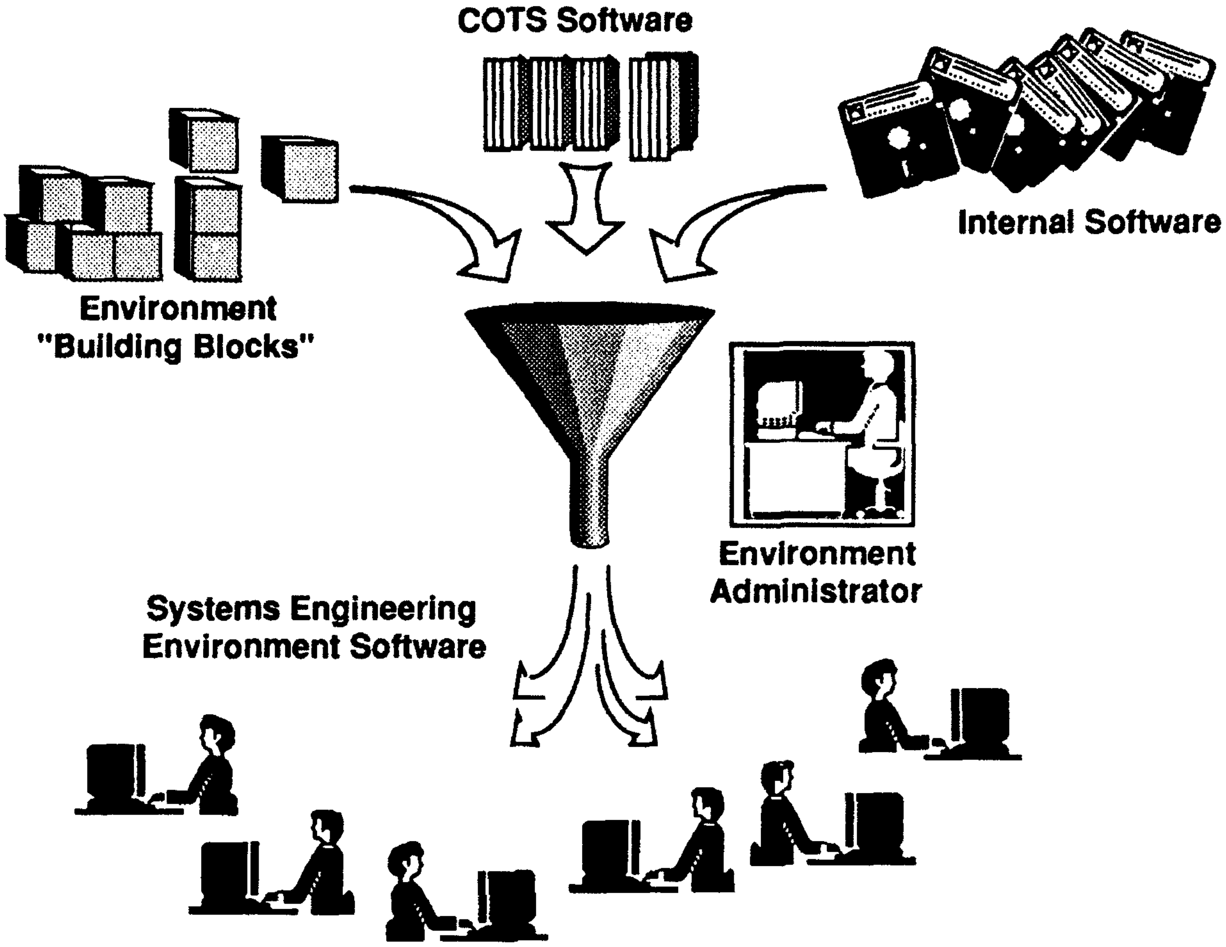

COTS Software
Internal Software
Environment "Building Blocks"
Environment Administrator
Systems Engineering Environment Software

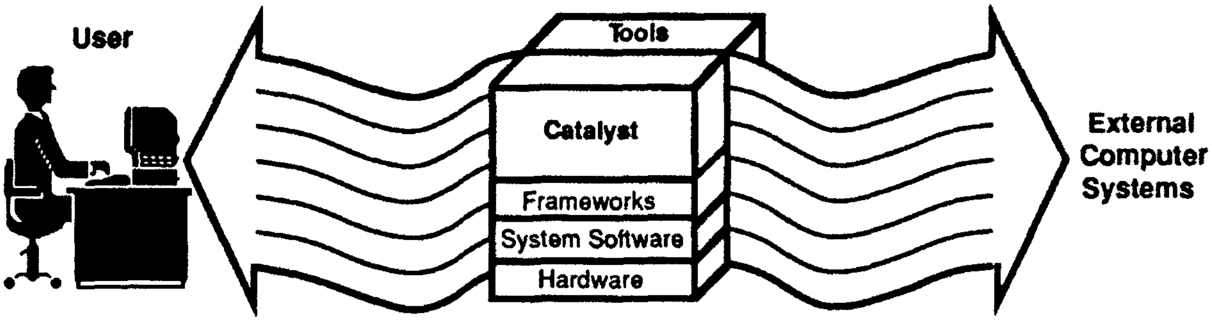
Figure 6.1-1. Building block concept for systems engineering automation

6.1.1 Catalyst Environment Interfaces

Catalyst has the following interfaces:

- User's host computer system, consisting of the computer hardware and system software
- User's installed frameworks
- User's installed tools
- Users (employing the services of the user's host computer system hardware, system software and possibly installed frameworks)
- External Systems (employing the services of the user's host computer system hardware, system software and possibly installed frameworks)

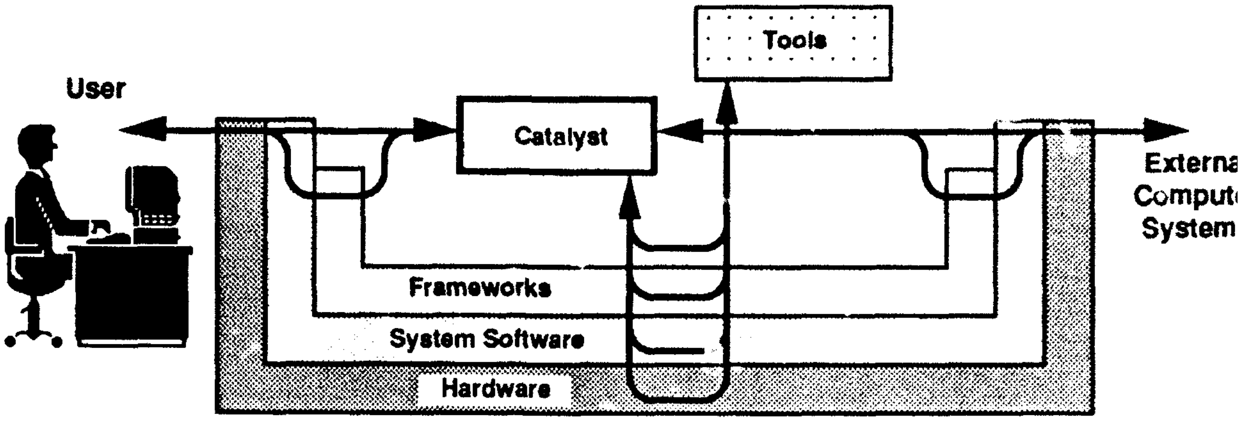
Note that the latter two interfaces are layered on top of the user's computer hardware, system software and frameworks. The program interfaces of the Catalyst software involve only the system software, user frameworks and user tools.



Communication between Catalyst and the user is accomplished through direct use of system software services (e.g., terminal drivers) or through use of user interface frameworks (e.g., X Windows, Motif), that in turn use the underlying system software and hardware to communicate with the user.

Similarly, communication between Catalyst and external systems may be accomplished through direct use of system software services (e.g., low level communication protocols such as X.25 or Ethernet) or through use of communication frameworks in the form of higher level communication protocols (e.g., NFS, TCP/IP), that in turn use the underlying system software and hardware to communicate with the external systems.

Catalyst communicates with other tools, either directly (e.g., procedural interfaces), through use of system software services (e.g., process-to-process message services), or through frameworks (e.g., tool access protocols). The underlying software services and frameworks typically provides facilities to communicate with tools that execute on distributed computer hardware in the system.



6.1.2 Computer System Environment Requirements

Catalyst must be adaptable to support the systems process across a wide range of computer system platforms and configurations, including the following:

- Standalone computer system with explicit communications to external systems
- Host-workstation configuration
- Heterogeneous computing network
- Multiple heterogeneous computing networks with internetwork communications

6.1.3 Environment Focus

The *Catalyst* automation focused on high payoff systems engineering automation, specifically in the areas of:

- **Modeling and specification meta-tools,** a set of highly adaptable and configurable tools that are used for a variety of tasks and in a number of different contexts, particularly those early life cycle requiremcnts and system design specification activities.
- **Concurrent engineering groupware,** tools specifically designed to promote and enhance multi-person, and particularly multi-disciplinary, interactions and collective work flows.
- **Integration mechanisms,** frameworks that increase the effectiveness of multi-tool work flows and promote information sharing between the systems engineer and the various specialty roles.

- **Environment administration support**, tools assisting the installation, adaptation and extension of the environment.

The determination of what capabilities were of highest payoff was determined from the market survey, process modeling and field interview activities.

6.2 Technology Demonstrations

As defined in the SECD SOW, the Technology Demonstrations identified tools and enabling technologies which were considered critical to establishing an automated system engineering environment. The following technologies were demonstrated for the SECD Project Team:

- FIELD (Friendly, Integrated Environment for Learning and Development), a system with message-based integration
- Versant, an object-oriented database product
- ArborText, an electronic document production product using SGML
- Automated Access Experiment (AAE), a SPS delivered system to RL for distributed, heterogeneous computer systems
- InQuisiX, a classification and search engine developed by SPS
- InSight, a SPS meta-tool
- Momenta, a Pen-Based Computer
- PRICE (Parametric Review of Information for Cost and Evaluation), a cost analysis tool

Each of these products represent an area of technology that is critical to the development of an automated systems engineering environment. The choice of these specific kinds of products to the SECD Project Team indicates their necessity to implement an operationally effective environment within the 5-7 year time frame.

6.3 Prototype Scenarios

The goals of developing a prototype of Catalyst were to sell the system concept and to reduce risk. These risks are many; development, performance, market, usability, maintainability, evolvability, feasibility, and unit cost. The prototype also demonstrated the availability of the technology within projected cost and schedule.

The prototypes demonstrated, through user scenarios, multi-disciplinary support for systems engineers and speciality engineers. Scenarios used realistic data and a combination of COTS tools and custom software. The prototype scenarios functioned on a heterogeneous network of two Macintoshes, a PC, and a Pen-based computer.

The prototype scenarios demonstrated the following concepts:

- Concurrent, multi-disciplinary engineering
- Requirements traceability and allocation, change management over multiple tools and disciplines
- Team support, groupware, work flow, meeting support
- Customized tools and methods from building blocks
- Process-knowledgeable environments
- Cooperative integration of COTS tools
- Heterogeneous network, PC-based tools

These concepts and features were combined into three demonstration scenarios: Requirements Flowdown, Timeline, and Tradeoff.

The Requirements Flowdown Scenario demonstrated requirements allocation and traceability, change impact analysis, document specification generation heterogeneous distributed computing, and the cooperative integration of COTS tools with a centralized database supporting the synthesis of information from multiple sources. This scenario used a data set from the integration of the Global Positioning System into the A-10A aircraft.

The Timeline Scenario demonstrated the ability to automate a custom method using meta-tool technology and groupware concepts supporting concurrent engineering. This scenario demonstrated automation of a mission analysis method used by the Naval Air Warfare Center, Aircraft Division, Warminster, PA.

The Tradeoff Scenario demonstrated a tradeoff harness for alternative analysis; individual tasking for multi-disciplinary analysis; interfacing to external analysis tools and mathematical solver tools; receipt of data from tools; synthesis and display of various analyses to support decision making; voice annotation (during tasking); and voice commanding (to view analysis results).

6.4 Evaluation of Rome Laboratory Software Engineering Facility

The Rome Laboratory Software Engineering Facility was evaluated and the following list of hardware and software was recommended to support Catalyst:

- A heterogeneous computing network using TCP/IP and GOSIP.
- At least one, each, of the most popular, UNIX/POSIX workstations. Currently, that includes Sun Microsystems (Sun), Hewlett-Packard (HP), Digital Equipment Corporation (DEC), and International Business Machines (IBM), as well as any other supported personal computers.
- At least one of each of the supported personal computer platforms, including Macintosh and x86 machines running Windows.
- A heterogeneous collection of popular general purpose tools (e.g., word processors, outliners, file makers, spreadsheets, drawing tools, presentation tools, and electronic mail).
- A collection of demonstration Commercial-Off-The-Shelf (COTS) analysis tools, system engineering tools, and relevant specialty engineering tools.
- Output devices that support the operating system and tools within the operational environment.
- Various emerging framework and object management packages.
- Facilities to demonstrate the automation of meetings and group work activities.

6.5 Risk Analysis

The analysis of the risk associated with SECD consisted of identifying the risk and planning a strategy to reduce those risks.

6.5.1 Risk Identification

We have identified the following risks for the *Catalyst* environment:

- **Scope** - Attempting to address a scope that is either too broad or too ill-defined is a risk to the successful specification and later implementation of the environment.
- **Usability** - The 1995 environment must be highly usable to insure technology transfer and marketability. The two key factors that contribute to usability are environment capabilities and user interface desirability and performance.
- **Performance** - Performance is the key to the usability and acceptance of the completed environment. However, performance requirements which are too stringent will add significant risk to the development project.
- **Degree of adaptability** - The environment must be sufficiently adaptable to successfully automate an organizations' existing processes. There is a

high degree of variability in the roles, methods, work products and work flows that must be accommodated.

- **Scalability** - The environment must be able to automate the development of systems involving hundreds or even thousand of personnel. Current environments have had significant problems in scaling up to support very large developments.
- **Feasibility** - Implementing the specified environment by 1995 is a risk that should be tracked closely during this contract. It is a trade-off against incorporating both "cutting edge" and state-of-the-art technology.
- **Maintainability** - One of the problems in marketing SLCSE is its maintainability. SLCSE is tightly integrated with a host of COTS software products (e.g., VMS, DECNET, SMARTSTAR, RDB/Sharebase) Any time one of the COTS components is upgraded, many hours of maintenance may be required to make SLCSE functional again. Our experience indicates that we must insure that the system engineering environment is easy to maintain and will suffer minimum impact by COTS component upgrades.

6.5.2 Risk Analysis and Abatement Strategies

Prudent planning and management dictated that potential problems were identified as early as possible and alternatives were enumerated to reduce the related program risk. The following analysis was accomplished according to the Air Force Systems Command and Air Force Logistics Command entitled, "Acquisition Management Software Risk Abatement," AFSC/AFLCP 800-45.

Four risk areas were identified:

1) Performance

2) Support

3) Cost

4) Schedule

The analysis process targeted software development. Support risks were not analyzed since were dependent upon the capabilities of the development contractor.

6.6 Cost Analysis

The following cost analysis was performed for Demonstration and Validation of Catalyst using man years of effort:

	Dem Val Cost (mnyrs.)	
CSCI	**Low**	**High**
01 Process Managemenl	25	50
02 Information Retrieval, Display and Editing	10	20
03 Meta-Modeling Toolkit	20	40
04 Modeling and Analysis	5	10
05 Work Product Productior	12.5	30
06 Catalyst Administration	30	60
07 Object Infrastructure	25	50
08 Common Object Service:	20	40
TOTAL	**147.5**	**295**

7. Summary of Security Study

FTR, Volume 7 - Security Study presents the details of the work performed by SECD subcontractor, Miguel Carrio, MTM Engineering, McLean, VA. The following three tasks were identified to better understand the security requirements automated systems engineering environment:

1. Assessment of Security Technology
2. Analysis of Catalyst Security Requirements
3. Recommendations for a Secure Catalyst Architecture

The assessment of security technology examined enabling technologies to allow practical use of Catalyst for secure applications. The technologies investigated focused on secure operating systems, trusted databases, communications, networking and host platforms.

The analysis of security requirements for Catalyst analyzed and established a core set of security requirements to enable Catalyst operation in a secure application. The core set of requirements were refined and matured throughout the design documentation hierarchy.

The recommendations for a security automated system engineering environment identified architectural techniques, views, approaches, and constraints to assist in the development of a viable Catalyst architecture. The tasks were short in nature and the SECD Project Team gained a sense of direction and availability of the current and promising technologies in security.

The sum of these tasks established an early infrastructure for the domain of security. Security and related issues are traditionally inadequately addressed or considered after the fact, with exception of intelligence agencies that must deal with these issues and sensitivities on a daily basis. The security study provided early insights into security technologies and issues a foundation for establishing a security protocol.

The implementation of a secure systems engineering environment requires the following items:

- A well formulated security scenario
- An initial set of resilient and "complete" security requirements
- A mature and validated SECD Process Model
- An integrated SECD model with other process models that can impact the overall system engineering process

- Identification of a security methodology
- Identification of a set of supporting security documentation
- Continued demonstrations utilizing the environment building blocks
- Identification of mission profiles and support activities
- Defined personnel roles and responsibilities

Initiation of the appropriate security activities in a timely manner is essential to the success of the SECD program. Resulting architectures will require time consuming assessments and evaluations, hence, the need to begin the security tasks early in SECD's life cycle.

In the final analysis, Government security requirements for trusted levels must be adhered to. The range of trusted levels required to protect information was identified in FTR, Volume 7 - Security Study, and took into consideration personnel security clearance requirements. Key conclusions are as follows:

- Levels of trust at the B1 level can be satisfactorily met by the marketplace.
- At the B2 level, for the Catalyst environment, the next 3-5 years appear to represent reasonable maturation times for additional product offerings.
- Achievement of an A1 level of trustedness, for the SECD environment diversity within the next five years, appears questionable and introduces a higher degree of risk.

A pragmatic approach to achieving these higher levels of trustedness (greater than B3) within this timeframe, would be to require supporting individuals to be cleared at the top secret level prior to joining an activity. The activity itself would require preparation for the higher clearance similar to what is done today for compartmented security.

In conclusion, with appropriately cleared individuals at the secret or top secret level; and data sensitivity at the same levels; the current state-of-the-practice, available technology and approaches satisfy many of the SECD requirements and mission roles. From a security programmatic view, if properly implemented, a secure Catalyst can be a reality.

8. National Council on System Engineering (NCOSE)

During the SECD effort, the SECD Project Team became active members of the National Council on System Engineering (NCOSE). NCOSE and its working groups provided a mechanism for networking with a selected community of over two hundred systems engineers representing major systems houses in defense and non-defense corporations across the nation. Visibility of the SECD effort and business relationships were actively pursued at the 1st Annual NCOSE Symposium and interim business and working meetings.

8.1 SECD Briefings to 2nd Annual International Symposium

As a result of the NCOSE memberships, the SECD Project Team participated in the 2nd Annual NCOSE International Symposium, July 20-22, 1992, Seattle, WA. The theme of the 2nd Annual International Symposium was "Systems Engineering in the 21st Century." Technical sessions, continuing education tutorials, and over 100 technical papers addressed four major topic areas: System Engineering Processes, Commercial System Engineering and Case Studies, Automation and Tools, and Education and Training. The SECD Project Team presented the following four technical papers in the Automation and Tools Session on July 21, 1992:

- Frank LaMonica, "System Engineering Concept Demonstration - An Overview"
- Richard Pariseau and Hank Stuebing, "Systems Engineering of Naval Air ASW Systems"
- Edward Comer, Catalyst: Automating Systems Engineering in the 21st Century"
- Alberto Ortiz, "System Engineering Concept Demonstration Process Model"

These four technical papers appear in Appendix A of FTR, Volume 1, and were published in the Proceedings of the 2nd Annual International Symposium for NCOSE, July 20-22. Seattle. WA. The four presentations were following by a question and answer period for the NCOSE members. These presentations provided additional visibility and public review for the work and demonstrations performed by the SECD Project Team.

8.2 SECD Demonstrations at Exhibition Hall

The SECD computer testbed and process model was also exhibited at the 2nd Annual International Symposium. Four demonstrations were given by SECD Project Team members which were the Requirements Flowdown Scenario, the Timeline Scenario, the Tradeoff Scenario, and browsing and discussion of the Process Model.

Figure 8.2-1 illustrates the layout of the Catalyst exhibit at the 2nd Annual International Symposium.

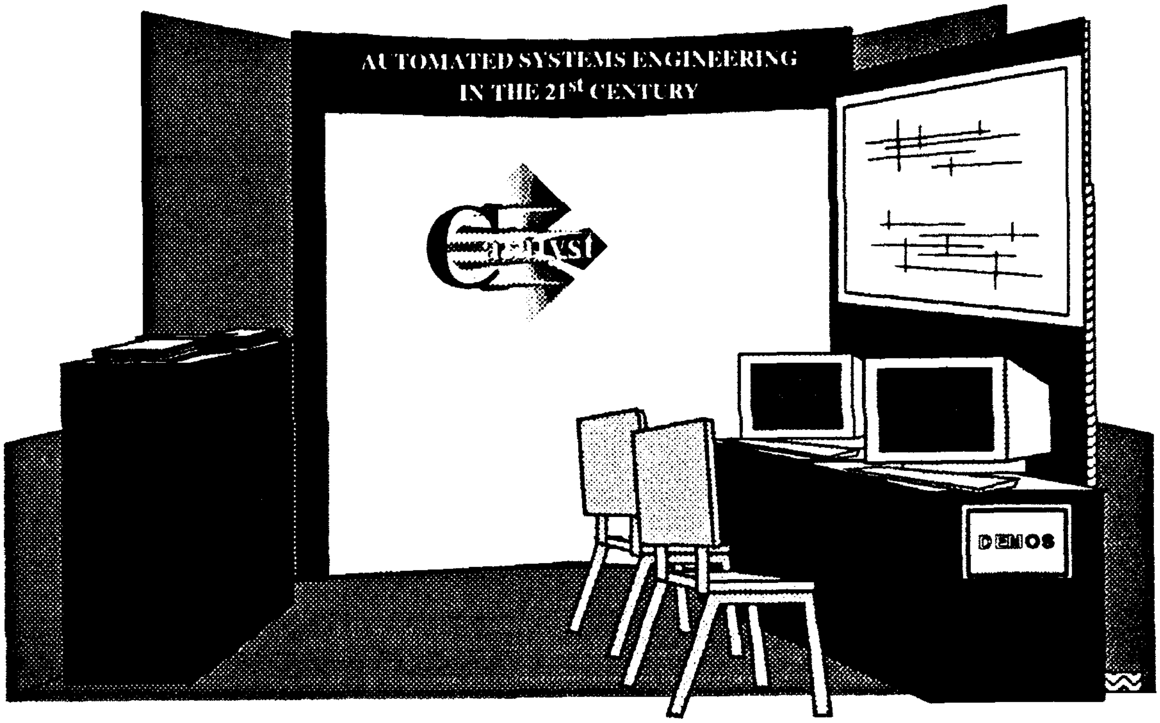


Figure 8.2-1 SECD Exhibit of Catalyst at NCOSE 2nd Annual Symposium

8.3 History of NCOSE

NCOSE resulted from a need to have a forum for discussion of system engineering issues, to improve the practice of system engineering, and to create a national voice for system engineering. A dozen local chapters should be activated by the end of 1992. NCOSE is a people movement by system engineers and system engineering managers that depends on the interest and energy of individuals to address critical issues.

At a time when there appears to be a growing need for new visions and new solutions the barriers of time and space are vanishing, according to the keynote speaker at the first NCOSE annual conference. Additionally, competition is increasing, information overload is prevalent, environmental problems are pervasive, and the problems facing America are becoming more complex. A new professional is needed that has the ability to think in terms of a "total" system, work as a member of a multi-disciplinary team, and solve the complex problems in an environment not constrained by the organizational and procedural support nets of the past. The National Council on Systems Engineering has the opportun:ty to be a leader in promulgating excellence in system engineering practices in industry, government, and academia, both DoD and non-DoD related. The synergism of strong local chapters, a strong national organization, and cooperative efforts with other associations will give system engineering a voice and the needed champions to determine our tomorrow.

NCOSE was created by system engineers for system engineers to:

1. Foster the definition, understanding and practice of world class systems, engineering in industry, academia, and government.
2. Provide a focal point for dissemination of system engineering knowledge.
3. Promote collaboration in system engineering education and research.
4. Assure the existence of professional standards for integrity in the practice of system engineering

In 1989 General Dynamics hosted a meeting of a few individuals at the University of California, San Diego campus to discuss the apparent shortage of qualified engineers that were able to think in terms of the total system rather than just their specific discipline, and could implement the system engineering process. They concluded that this problem was a national rather than a local problem and agreed to recruit more representatives of government, industry, and academia to examine this issue.

The next meeting was hosted by Boeing at the Battelle Conference Center in Seattle during the Summer of 1990. Over 30 individuals attended this meeting. Recognizing the major differences in views on the definition of system engineering, the meeting organizers grouped the participants with similar views on system engineering into three separate groups. Surprisingly, all three groups identified similar issues. These issues and concerns are recorded in Table 1.

The group agreed that a national organization was needed to address these concerns and defined the charter, and an ad hoc structure for NCOSE. Six committees were formed to address (1) the process of system engineering, (2) case studies and benefit cost studies, (3) university activities, (4) industrial training efforts, (5) Defense System Management College (DSMC) interactions with NCOSE, and (6) system engineering process maturity. Harry Carlson from Lockheed, Jerry Lake from DSMC, and Brian Mar from the University of Washington were selected as the provisional co-chairpersons of NCOSE. These three, with the help of council supporters, organized and lead the activities of the organizational meetings during 1990 and 1991.

The Aerospace Corporation hosted the January 1991 NCOSE meeting in Los Angeles. Over 60 individuals attended this meeting. In addition to working sessions, a few selected papers were presented to provide detailed views of different system engineering issues. The initial six committees were reorganized into (1) a Communications Committee to address the transfer of knowledge through various media including national conferences, (2) a Systems Engineering Practices Committee to address process and practice and (3) a Systems Engineering Development Committee to address education, training, and certification issues. The issues identified in Table 1 continued to be addressed with a focus on defining the process and the education and training of system engineers. A steering group was created to address the administrative issues associated with the institutional issues of formalizing NCOSE as an incorporated council open to national membership. The major outcomes of this meeting were (1) a decision to hold the first national open meeting, (2) to also hold an academic workshop, (3) to develop a position paper on the proposed MIL-STD-499 revision, and (4) to incorporate NCOSE.

IBM sponsored the first academic workshop held in Rockville, Maryland during June, 1991. This academic workshop was organized by Odd Asbjornsen of the University of Maryland. Following a day-long presentation of papers describing different university system engineering programs and in-house training programs, a one day workshop was held to address (1) the definition of system engineering as a process and the definition of a professional profile for a systems engineer, (2) tools and computer aids that support system engineering practice and education, and (3) design of system engineering curricula at various levels. The proceedings of this work-shop are available from Ginny Lentz at IBM.

Immediately after the academic workshop TRW hosted a business meeting of NCOSE in Alexandria, Virginia to plan the first annual conference, to continue committee work, and to continue incorporation. A major issue addressed was the review and critique of the proposed changes to MIL-STD-499, and the drafting of a NCOSE position paper on the standard. NCOSE provided a neutral forum for discussion of the 499 proposals since the authors of the proposed changes as well as government and industry representatives on the 499 steering group are members of NCOSE. Proposed changes using the system engineering management plan (SEMP) to define the contractual work and the lack of a higher level coordinating organization to resolve conflicting standards were of major concern. The communication group selected papers for presentation at the First Annual NCOSE conference to be held in cooperation with the American Society for Engineering Management (ASEM) in Chattanooga, Tennessee during October of 1991. The Development Group completed surveys of undergraduate and graduate system engineering programs, employer requirements for system engineers, and the desired personal characteristics of a qualified systems engineer. The surveys revealed that less than 20 universities offered one system engineering undergraduate course, and only a few offered two. Most of these courses addressed simulation and optimization rather than the system engineering process. The NCOSE administrative efforts were formalized to address the issues of incorporation, membership, election of officers and ways and means.

The first annual conference of NCOSE was held during October, 1991 in conjunction with ASEM in Chattanooga, with attendance of over 100 individuals registered for the NCOSE meeting. The paper presentations addressed the practice, process, and tools supporting system engineering. Papers were published in the conference proceedings.

During the business meeting that followed the conference, Seattle was selected as the location of the second annual conference to be held during July of 1992. The newly formed Seattle local chapter of NCOSE volunteered to host this conference. Larry Pohlmann, Boeing, presented a comprehensive report of the Practices Committee featuring the use of the system engineering process to develop committee tasks and products. groups of four to eight individuals were assigned to address specific issues of concern. Subgroup issues included the continued review of policy and standards, the definition of best practices, the search for better automation tools, the identification of pragmatic principles, and the search for metrics. At this meeting the need to diversify the focus within NCOSE to include non-defense sectors was recognized. A major effort was defined to include commercial, governmental, and industrial sectors outside of the defense and space sectors in future NCOSE activities. A call for papers for the Seattle meeting was distributed, a slate of candidates for officers was adopted, and the final steps of incorporation were approved.

Hughes hosted the January, 1992 business meeting in Los Angeles with NCOSE now a formally incorporated organization. Barney Morais, Synergistic Inc., headed the incorporation effort. The first set of elected officers were Harry Carlson, past president, Jerry Lake, president, Brian Mar, president elect, Barney Morais, treasurer, and Jeff Grady from General Dynamics, secretary. The six directors elected were Jim Brill from Hughes, Jim Cloud from Motorola, Dave Clemons from General Dynamics, George Friedman from Northrop, Jim Lacy from Texas Instruments, and Wayne Wymore from SANDS.

At this meeting the functional committees met one day and the administrative committees met the second day to permit all members an opportunity to participate in the functional issues. This arrangement was well received and will be continued in future meeting. The Practices Group addressed the issue of requirements management and metrics, and the Development Group identified five critical issues related to improving the education and training of system engineers. Administrative issues included the coordination of local chapters, the recruitment of new members, and ways and means issues.

Other results from this meeting were: (1) The first petition for a local chapter was approved. This was the Seattle chapter which reported an enrolled membership of almost 100. It was announced that other local chapters were being formed in San Francisco, southern California, Arizona, Texas, St. Louis, Cleveland, and Washington, D.C. This indicates the strong interest there is in NCOSE. (2) In response to the call for papers for the Seattle 1992 meeting, over 100 abstracts were submitted., The review of these abstracts involved many individuals. Over 80 papers were selected for presentation.

8.4 NCOSE Issues

At each NCOSE meeting, an executive of the host organization presented a keynote address identifying their vision of system engineering. These presentations revealed major differences in views of system engineering. The following section summarizes the different concerns expressed by participants in NCOSE meetings to date. Most of these also appear in Table 8.1-1. Despite these differences, the common need to create a national forum and voice for system engineers grows. It is important for new members to recognize and respect these differences. This section identifies some of the legitimate on-going differences that are represented by the NCOSE membership.

Table 8.1-1 Representative issues identified at the Seattle Organizational Meeting

GENERAL How do you define system engineering? What is a systems engineer? Who is a good systems engineer? Is it system or system engineering? Difference between system engineering, design, integration and management? System engineering lacks society, journals, and stature.	PROGRAM RELATED Systems engineering is a people problem. Systems engineering not used to manage the program, only the product. Administrators and managers do not realize need for system engineering. Too much specialization, specialty integration poor. Relationship between system engineering, concurrent engineering and TQM.
PROCESS RELATED Dealing with complexity is a difficult task. Products need more optimization. Is system engineering a philosophy or a discipline? What is the role of system engineering in a product life cycle? Benefits of system engineering needs to be documented How do you measure the health of the system engineering process? Process may be application specific.	TRAINING RELATED Is there a shortage of systems engineers? Who should be trained? Heuristics difficult to learn. Need to train how to function as a team member. Lack of cross functional education.
PRODUCT RELATED Need to optimize requirements, designs, and products. How to manage and trace requirements. Operational concepts poorly specified. Operational concepts lacking. Lack of tools and automation to support requirements development.	ACADEMIC RELATED Lack of clear requirements for a system engineering degree program. Few system engineering graduates. Few accredited system engineering programs. Lack of textbooks on system engineering. Should all engineering students take a system engineering course? Systems engineering oriented programs have different names.

PROGRAM VERSUS PRODUCT One of the early differences addressed by NCOSE is the definition of system engineering and the role of system engineers in defining and developing products such as missiles, spacecraft, and weapon systems. Some members view system engineering as a technical activity that defines the architecture and requirements for the design of new products. Under this view system engineers perform requirements analysis and trade studies in order to generate specifications for the system. An opposing view is that system engineering involves the management of a process that defines the tasks and activities needed to create a product. This latter view considers system engineering to be a process rather than a discipline. The proposed changes to MIL-STD-499 focuses on the system engineering processes of a system through the use of cross-functional teams. It is this process that must be managed. This difference between a process management focus and a product architecture focus reflects the job orientation of different NCOSE members and suggests that system engineering can be applied to process development as well as product development activities.

FRONT END VERSUS BACKEND Another major difference in views of individuals attending the NCOSE meetings is whether the basic task of system engineers is to translate customer's needs into specifications that can be allocated to designers and contractors, or whether the system engineering function continues throughout the life cycle and includes the "ilities". Again the job orientation of an individual may cause them to focus their system engineering on a particular phase of the life cycle, but this spectrum of applications suggests that the system engineering process can be applied to each phase of the life cycle.

SYSTEM ENGINEERING VERSUS CONCURRENT ENGINEERING The concept of system engineering has existed for decades. More recently total quality management (TQM) and its concurrent engineering initiative have become popular. Do these two activities have common elements? Proponents of each concept tend to have different views. System engineers tend to argue that their structured process of functional analysis, requirement allocations, and verification of designs and products are basic for total quality management and an effective process to facilitate concurrent engineering. Proponents of concurrent engineering seek to improve communication between specialist by requiring parallel definition of products and processes employing cross-functional teams and a system engineering methodology. Both paradigms seem to require the structure and traceable problem definition advocated by the system engineering process.

SYSTEMS ENGINEER VS. ARCHITECT Many organizations seek system architects who can conceive new product concepts. A system architect creates new systems in the same manner that a conventional architect creates new buildings. These architects are viewed as the elite few individuals in a firm who define the products of the future. Whether these individuals embrace the system engineering process or not does not seem to be an issue. System engineers argue that it is the senior systems engineer who is the architect, while others argue that knowledge of system engineering is not required to become an architect.

SYSTEMS ENGINEER VS . SYSTEMS ANALYST Most universities are developing systems analysts rather than system engineers. These analysts use simulation and optimization tools to describe systems and define optimal architecture. They lack the ability to translate customer needs into meaningful requirements or to provide structure and discipline to the development process. Much of the concern expressed by practicing system engineers tends to focus on the lack of training or education in new graduates to perform functional and requirement analysis associated with acquisition engineering.

DEFENSE VS. COMMERCIAL PRACTICE While NCOSE was created by individuals involved in defense or space activities, participants at recent NCOSE meetings have emphasized the need to expand the NCOSE membership to include individuals practicing system engineering in commercial enterprises. There have been participants from communications, energy and commercial airplane programs, and a few from the automotive industry. Participants from the private and governmental sectors associated with consumer goods, food products, and urban infrastructure are now being recruited. Systems engineering is a process that should be applied to the development of any complex product line or program.

8.5 NCOSE Programs

The products generated by NCOSE are being formalized in working group reports, electronic bulletin boards, conference proceedings, etc. The informal exchange of information associated with personal networking provides an even more valuable product. Each working group has a growing list of committee reports. The Practices Committee has produced position papers on the proposed MIL-STD-499 revision, best practices, process descriptions, tools and automation, pragmatic principles, and metrics to name a few. The Development Committee has produced surveys of system engineering programs in universities, profiles of system engineers, and proceedings of the first academic workshop containing over a dozen papers. The Communications committee has established annual technical conferences and has created an electronic bulletin board where newsletters, directory of members, and other information can be accessed.

9. Conclusions of SECD

System engineering is a broad, highly variant discipline that is critical to the successful development and support of mission-critical systems. The needs analysis performed under this effort has identified a host of problem areas in the state-of-the-practice of system engineering that need attention.

Automated solutions to those problems have the potential to significantly increase the effectiveness and reduce the cost and risk associated with system engineering activities throughout the system life cycle. Catalyst technology has been carefully thought out to effectively address those needs and to also be amenable to an incremental technology transfer approach which is imperative to its successful deployment.

The feasibility, usability, and marketability of Catalyst has been fortified by the various trade off studies and analyses, and concept and technology demonstrations that were performed. Catalyst will enhance the effectiveness of the system engineering process to produce more successful, higher quality systems. SPS recommends the RL R&D program should strive to continue the development of Catalyst technology for effective and productive system engineering.

10. Areas for Future Research and Development

The next steps for future research and development envisioned for the next seven to ten years includes parallel activities in the following four areas:

1. Development of a functionally-complete Advanced Development Model (ADM) of Catalyst through a series of incremental prototypes and builds, developing and demonstrating state-of-the-art automated system engineering technologies.
2. Integration and demonstration of B2-level components for a secure Catalyst configuration.
3. Continued research into a number of the emerging technology areas.
4. Incremental user review, assessment and experimentation with the various Catalyst technologies providing feedback into the continued research and development.

The following section address each of these four areas.

10.1 Catalyst Advanced Development Model

The objective of the Catalyst ADM will be to incrementally produce a toolset of sufficient quality and robustness to enable experimentation by systems engineers and specialty engineers. Because of the broad scope of the Catalyst requirements stated in the Catalyst SSS, the process of incrementally developing the Catalyst ADM will likely take the remainder of the decade.

It is recommended that the initial major Catalyst build should focus on the following:

1. Application of a new object-oriented message-based integration approach. Object management and messaging facilities would be acquired commercially. Early demonstration of the core Catalyst integration facilities is critical for the future developments.
2. Development of a modeling meta-tool facility to support a variety of industry "best practices" for engineering computer-intensive applications and to provide an level of end-user tailoring. Support for semi-formal methods was identified as a major gap in COTS automation.

In order to provide a significant set of initial capabilities, a variety of emerging commercial-off-the-shelf (COTS) capabilities should be leveraged and integrated for project management, editors and browsers, data display and analysis, documentation, electronic communications, and information retrieval. The Rome

Laboratory E-SLCSE can likely be leveraged in the areas of process management, documentation and impact analysis.

Additional capabilities would be developed and integrated in subsequent builds based upon the results of user feedback and priorities, the other parallel security and research tasks, and based upon funding availability.

10.2 Secure Catalyst

Automation of the system engineering process will require a secure Catalyst configuration. The area of multi-level secure technologies are emerging to the point where a meaningful integration and demonstration of a secure Catalyst is feasible within the next five years. The development of a secure Catalyst will require the following:

1. Integration of emerging B2-level components for operating system, object management, windowing system, and communication on a trusted computing base (TCB).
2. Layering and adaptation of selected Catalyst automated facilities onto the secure base.
3. Development of secure Catalyst configuration management, administration and operation policies and procedures.
4. Development of automated security administration tools.

It is recommended that a secure Catalyst demonstration be pursued in parallel with the initial Catalyst ADM builds because of the significant risks associated with the multi-level security problem. Once a secure configuration is successfully demonstrated, the Catalyst ADM development should be upgraded to support secure operation.

10.3 Research Areas

The SECD effort highlighted a number of emerging technologies that warrant additional research in the near term in order to mature them for eventual incorporation into later builds of the Catalyst ADM. Such areas would be appropriate for 6.2, Small Business Innovation Research (SBIR), or in-house efforts.

Recommended research areas include the following:

- **Process and task enactment in a process-knowledgeable environment.** Specific issues include the development of large-scale process models, the tailoring of process model templates, the enactment of processes at various levels from high level phases through low level tasks, and user interface paradigms for acceptable process enforcement.

- **New user interface paradigms for user workspaces.** The SECD effort conceived a workspace concept to enable micro-level team tasking and support a high degree of user multiplexing. Continued research and prototyping is needed to solidify this concept.
- **Application of new alternative-media technologies to system engineering.** New emerging alternative-media technologies seem to have great potential for system engineering, including handwriting capture and recognition via pen-based computers; voice capture, playback and recognition; electronic signatures and personal identification devices; video capture, playback and editing; image scanning and document recognition; and multi-media application composition and execution systems.
- **Systems engineering metrics.** New metrics are needed for system engineering. Such metrics would need to include interim process measures and metrics on informal work products. Goal-oriented and risk-oriented metrics should be investigated. Specialty engineering metrics are needed to better enable systems engineers' monitoring of specialty activities.
- **Groupware for system engineering.** Emerging "groupware" concepts should be explored for their applicability to team-oriented system engineering processes that occur with a concurrent engineering approach. Topics include include automated approaches for team-oriented creativity and exploration, multi-person problem solving, organizational decision making, electronic meetings, notification and awareness, group process planning and enactment, and tasking and monitoring paradigms.

10.4 User Participation

Continuous, in~remental user review, assessment and experimentation with the Catalyst technologies is critical to the ultimate success of the program. Various multi-year contractual and in-house programs will develop critical Catalyst components and will use those components to incrementally assemble the Catalyst advanced development model. The Catalyst ADM will be used for demonstration and experimentation purposes. Success of the ADM will determine future transition emphasis to an engineering development phase.

Specific recommendations for user involvement include the following:

- **Core review team from potential user organizations.** Because system engineering is, today, largely a manual process, continuous user involvement is important. The SECD effort profited from a core review team that represented both government and contractor organizations. Specific individuals were committed from the kickoff review and participated throughout in quarterly reviews. This approach should be continued.

- **User assessments of early user concepts and prototypes.** Early prototypes, demonstrations and user concepts should be incrementally assessed by candidate users. Early prototyping is recommended for all key user concepts.
- **Realistic experiments using Catalyst builds.** Substantial, cohesive builds should be transitioned to experimental usage by real system engineering teams on real system engineering problems. "Shadow" tasks are recommended for early Catalyst experiments to minimize risks to government or contractor programs.
- **Periodic public review and participation.** The SECD effort utilized meetings and the conference of the National Council on System Engineering (NCOSE) for public review. The NCOSE participants were exactly the right audience for such an activity. Continued liaison with NCOSE is recommended.

Appendix A. NCOSE Technical Papers

The following technical papers were written for presentation at the 2nd Annual International Symposium of the National Council on Systems Engineering, held in Seattle, Washington on 20-22 July 1992 .
The first paper, written by Mr. Frank LaMonica from the U.S.A.F. Rome Laboratory, provides an overview of the SECD effort, including its goals, objectives, and results. The second paper, written by Mr. Edward Comer from SPS Inc., describes the systems engineering environment, given the name *Catalyst*, that was ultimately specified. The third paper, written by Mr. Alberto Ortiz from MDC-DAC, describes the development of a generic life cycle process model for systems engineering which was used to 1) demonstrate coverage of the systems engineering process by the *Catalyst* environment requirements and 2) provide a "setting" for the various concept demonstrations produced. The final paper, written by Mr. Richard J. Pariseau and Mr. Henry G. Stuebing, both from the Naval Air Warfare Center Aircraft Division Warminster, describes the results of the interviews with systems engineers at the facility and introduces the concept of their time-line methodology which was ultimately incorporated into one of the concept demonstration scenarios.

Order of Papers:

System Engineering Concept Demonstration– An Overview
Catalyst: Automating Systems Engineering in the 21st Century
System Engineering Concept Demonstration (SECD)– Process Model
System Engineering of Naval Air ASW Systems

SYSTEM ENGINEERING CONCEPT DEMONSTRATION - AN OVERVIEW

Frank S. LaMonica
Rome Laboratory
Bldg 3, M/S C3CB
Griffiss Air Force Base NY 13441

Abstract. This technical paper provides an overview of Rome Laboratory's Research & Development Program in the area of System Engineering. In particular, it describes the objectives of an exploratory development effort entitled "System Engineering Concept Demonstration" (SECD), the activities that were pursued under the effort, and its results. It serves as an introduction to other papers [Comer 1992], [Ortiz 1992], and [Pariseau & Stuebing 1992], which provide further detail on the technology and demonstrations that resulted. Rome Laboratory's future plans to develop Systems Engineering technology are also identified.

BACKGROUND

Rome Laboratory, formerly the Rome Air Development Center (RADC), became one of the four Air Force "super" laboratories as a result of the 1990 Laboratory and Research Center Realignment within Air Force Systems Command (AFSC). Located at Griffiss Air Force Base NY, Rome Laboratory is the designated Air Force laboratory for Command, Control, Communications, and Intelligence (C3I) technology. Four mission directorates exist within the Rome Laboratory organization, supporting research & development (R&D) in the areas of Command, Control, & Communications (C3), Surveillance & Photonics, Intelligence & Reconnaissance, and Electromagnetics & Reliability.

System Engineering Concept Demonstration (SECD) was an exploratory development effort sponsored and managed by Rome Laboratory. The SECD contract was awarded in February 1990 to Software Productivity Solutions (SPS) Inc., Indialantic FL, as the prime contractor. McDonnell Douglas Corporation - Douglas Aircraft Company (MDC-DAC), Long Beach CA, and MTM Engineering Inc., McLean VA, were subcontractors.

As a result of a joint U.S. Air Force - U.S. Navy Memorandum of Agreement, personnel from the Naval Air Warfare Center Aircraft Division Warminster collaborated with Rome Laboratory and contractor personnel on the effort. Also, Dr. Walter R. Beam, a reputable consultant in the Command & Control (C2) arena, was funded by Rome Laboratory to perform several related systems engineering studies and helped to ultimately scope the effort and shape its technical direction and results.

INTRODUCTION

System engineering is one of the most critical aspects of successful, large scale, software intensive mission-critical system acquisition, development, and post-deployment support. It is a process that commences with the earliest phases of the system life cycle and continues until the system is retired from use. Regardless of the application domain, engineering large software intensive systems is a complex activity, typically involving hundreds, if not thousands, of geographically distributed engineers representing numerous specialties (e.g. hardware, software, human factors, mission analysis, sensors, supportability, reliability & maintainability, producibility, etc.).

The system engineer has the critical role - being responsible for transforming user and/or mission requirements into a real, cost-effective system. A need exists to enhance the effectiveness of the systems engineering role, the numerous supporting specialty engineering roles, and the collaboration which takes place between them.

The overall goal of the SECD effort was to increase the productivity and effectiveness of systems and specialty engineers involved in the development, maintenance, and enhancement of military computer-based systems. To effectively accomplish this goal, the effort was scoped as follows:

o It addressed the *systems engineering process* rather than the entire *engineering of systems process*. The *engineering of systems process* is considered a very broad and comprehensive one which involves numerous specialty engineering, analysis, assurance, and management roles that span the entire system life cycle. Systems engineering, on the other hand, is a single role that serves as the glue or catalyst between these various disciplines.

o It addressed *automation of the systems engineering role* and various activities from specialty roles that support systems engineering.

o It focused on *high payoff* automation rather than attempting to provide comprehensive coverage for all of the systems engineering automation throughout the life cycle.

o It focused on the more *generic* systems engineering functions and sought to interface or adapt to other areas that are specific to the type of application, the level of system being engineered, and organizational peculiarities.

The name *Catalyst* was selected to refer to the systems engineering automation objective of this effort - reflecting a view that the systems engineer and his/her associated systems engineering automation serve as the catalyst for the various specialty roles required in developing a system. Just as a chemically-based catalyst is a substance that modifies and increases the rate of a chemical reaction, *Catalyst* enhances the capabilities of systems and specialty engineers and provides an effective environment for increased and highly efficient interaction between them.

To accomplish the goal of this effort, a number of objectives were established which include: 1) the development of an operational concept for a systems engineering environment, 2) the specification of system requirements and system design for the environment, and 3) the demonstration of advanced concepts as well as enabling technologies deemed necessary to implement an operationally-effective environment within the 5-7 year time frame.

RESULTS

The SECD effort was comprised of trade-off studies and analyses, document/specification development, and concept and technology demonstrations.

The trade-off studies and analyses included a market analysis which consisted of a survey of studies and reports describing documented problems during mission-critical systems development and maintenance in order to identify common systems engineering problems and issues. This analysis was augmented by several field interviews that were conducted with practicing systems engineers at Rome Laboratory, Naval Air Warfare Center Aircraft Division Warminster, and IBM (Owego NY). The interviews focused on: 1) understanding the areas and degrees of variability in systems engineering processes, and 2) identifying areas of high priority need for systems engineering automation. Technically-oriented areas of need that were identified include requirements engineering, collaboration between systems and specialty engineers, system and subsystem interfaces, risk management, change and complexity management, quality engineering and assurance, and integrated support environments.

The trade-off studies and analyses also addressed hardware/software computing technologies, existing and emerging interface standards, environment framework technologies, and multi-level security needs and implementation possibilities. In addition, a generic process model was produced to document and demonstrate the breadth of the systems engineering process and insure that important systems engineering activities and needs were not overlooked.

The concept and technology demonstrations, which focused on risk abatement, provide feasibility demonstrations for potential users. Specifically, the concept demonstrations focused on prototype systems engineering concepts and technologies which are implementable in the 5-7 year time frame. Conversely, the technology demonstrations focused on viable, enabling technologies from both cost and performance aspects that have the potential for use in implementing a systems engineering environment within the next 5-7 years.

Catalyst is envisioned as a set of highly adaptable and configurable software "building blocks" consisting of interactive systems-oriented software tools, interface mechanisms, and environment frameworks. When integrated with an organization's installed computer system (i.e. hardware and operating system software), and other organization-specific tools and integrating frameworks, these building blocks will provide enhanced automated support for systems engineering. Organization-specific tools and integrating frameworks may include support for mission prototyping, modeling and simulation, cost analysis, reliability and maintainability, software engineering, hardware design and manufacturing, integration of hardware and software, etc. (ref figure 1).

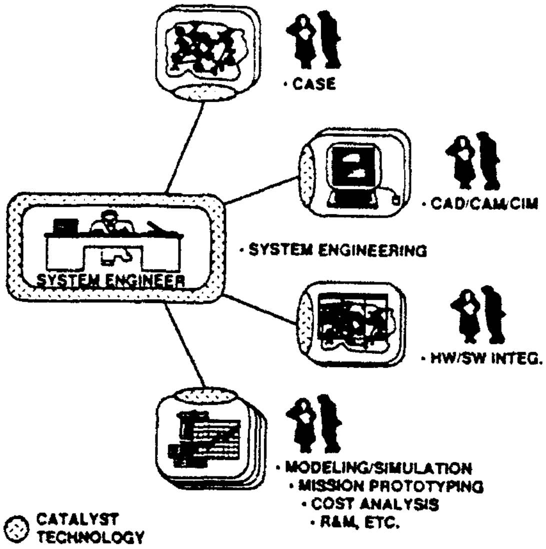


Figure 1: System View Of *CATALYST*

Catalyst focuses on high payoff systems engineering automation specifically in the areas of:

Modeling and specification meta-tools - a set of highly adaptable and configurable tools that may be used for a variety of tasks and in a number of different contexts, particularly those early life cycle requirements and system design specification activities. The intent of the meta-tools is not to be able to rapidly duplicate the capabilities of an existing commercial-off-the-shelf (COTS) tool, but rather to give systems and specialty engineers the capability to quickly configure a custom tool capability, that is not readily available, for his/her particular and immediate needs.

Concurrent engineering groupware - tools specifically designed to promote and enhance multi-person and, in particular, multi-disciplinary interactions and collective work flows.

Integration mechanisms - frameworks that increase the effectiveness of multi-tool work flows and promote information sharing between the systems and specialty engineers.

Environment administration support - tools that assist in the installation, adaptation, and extension of the systems engineering environment.

The building block approach for *Catalyst* was pursued, in lieu of attempting to specify a single, monolithic systems engineering environment solution because of several reasons: 1) the shear breadth of systems engineering, as practiced across the numerous types of mission-critical systems and application domains, would be difficult to specify and even more difficult to afford, 2) the high degree of variability in systems engineering processes across organizations precludes a single environment solution, 3) most organizations have experience with and depend on an installed base of tools, frameworks and computing hardware that would be difficult to quickly change, 4) to be successful, a technology transition of significant size must be incremental and must not negatively impact the user's day-to-day production activities, and 5) a single monolithic environment is not considered commercially viable; organizations generally prefer to select tool and environment technology components from multiple sources.

The feasibility, usability, and marketability of *Catalyst* has been fortified by the various trade-off studies and analyses, and concept and technology demonstrations that were performed.

Catalyst technology is described in greater detail in [Comer 92].

Documentation. Five documents/specifications have been produced, including a System/Segment Specification (SSS), System/Segment Design Document (SSDD), Operational Concept Document (OCD), Interface Requirements Specification (IRS), and Final Technical Report (FTR). The SSS, SSDD, and IRS were produced in accordance with DoD-STD-2167A data item descriptions. The OCD was produced in accordance with the American Institute of Aeronautics and Astronautics (AIAA) Recommended Technical Practice: Operational Concept Document (OCD) Preparation Guidelines (approval pending). The FTR provides a summary of the effort and has six supporting volumes associated with it. These include volumes entitled: 1) Systems Engineering Needs, 2) Process Model, 3) Interface Standards Studies, 4) Technology Assessments, 5) Trade Studies, and 6) Security Study.

Process Model. The process model is important for three reasons. First, it serves to provide a better understanding of the breadth and scope of system engineering activities that take place within the military system life cycle. Secondly, it was used to verify the completeness of the *Catalyst* system requirements. Finally, it is a "generic" model which can conceivably be tailored or adapted to meet a specific organization's needs.

The process model is implemented and demonstrable on a Apple Macintosh II Workstation using the COTS

MacFlow product. The approach and inputs used to develop the process model are described in [Ortiz 92].

Concept Demonstrations. Results of the effort identified seven needed features/concepts to be demonstrated:

o Concurrent, multi-disciplinary engineering involving team-oriented interaction and joint work products.

o System partitioning, requirements traceability and allocation, and change management.

o The meta-tool concept illustrating the automation of custom, semi-formalized methods.

o Process-knowledgeable environments.

o Cooperative integration of COTS tools.

o Heterogeneous network integration and access.

o Automated document/specification generation.

The features/concepts were combined into the following three demonstration scenarios: Timeline, Tradeoff, and Requirements Flow Down.

Timeline Scenario. The timeline scenario demonstrates 1) the ability to automate a custom method using meta-tool technology, and 2) groupware concepts supporting concurrent engineering which include electronic meeting rooms for reviews and remote electronic conferencing with audio link and concurrent display of screens remotely for joint work product development or review. Specifically, this scenario demonstrates automation of a mission analysis method used by the Naval Air Warfare Center Aircraft Division Warminster. [Pariseau & Stuebing 92] provides a basis for this scenario.

Tradeoff Scenario. The tradeoff scenario demonstrates a tradeoff harness concept for alternative analysis, individual taskings for multi-disciplinary analysis, the ability to interface to external analysis tools and mathematical solver tools and receive data from them, synthesis and display of various analyses in order to support decision making, and voice annotation (during tasking) and voice commanding (to view analysis results).

Requirements Flowdown Scenario. The requirements flowdown scenario demonstrates requirements allocation and traceability, change impact analysis, document/specification generation, heterogeneous distributed computing, and the cooperative integration of COTS tools with a centralized database supporting the synthesis of information from multiple sources.

FUTURE PLANS

Rome Laboratory is actively pursuing the transition of SECD exploratory development results to an advanced development program in system engineering, specifically to develop and demonstrate state-of-the-art system engineering technologies - including supporting tools and life cycle environments.

The planned multi-year contractual and in-house program will develop critical *Catalyst* components and will use those components to assemble an advanced development prototype of a systems engineering environment. The environment will be used for demonstration and experimentation purposes. Success of the advanced development prototype will determine future transition emphasis to an engineering development phase.

CONCLUSIONS

Systems Engineering is a broad, highly variant discipline that is critical to the successful development and support of software intensive mission-critical systems. The needs survey performed under this effort has identified a host of problem areas in the state-of-the-practice of systems engineering that need attention. Solutions to those problems have the potential to significantly increase the effectiveness and reduce the cost and risk associated with systems engineering activities throughout the system life cycle. *Catalyst* technology has been carefully thought out to effectively address those needs and to also be amenable to an incremental technology transfer approach which is imperative to its successful deployment.

The various analyses, trade studies, and demonstrations that have been prepared support the feasibility of the *Catalyst* concept. The Rome Laboratory R&D program will strive to continue the development of *Catalyst* technology for effective and productive systems engineering.

REFERENCES

Comer, E.R., "Catalyst: Automating Systems Engineering in the 21st Century", *Proceedings of the Second Annual International Symposium of the National Council On Systems Engineering (NCOSE)*, Seattle WA, 20-22 July 1992.

"Data Item Descriptions for DoD-STD-2167A", Defense System Software Development, 29 February 1988.

"Operational Concept Document (OCD) Preparation Guidelines", American Institute of Aeronautics and Astronautics (AIAA) Recommended Technical Practice (approval pending), 1 March 1992.

Ortiz, A., "System Engineering Concept Demonstration (SECD) Process Model", *Proceedings of the Second Annual International Symposium of the National Council On Systems Engineering (NCOSE)*, Seattle WA, 20-22 July 1992.

Pariseau, R.J. and Stuebing, H.G., "Systems Engineering of Naval Air ASW Systems", *Proceedings of the Second Annual International Symposium of the National Council On Systems Engineering (NCOSE)*, Seattle WA, 20-22 July 1992.

"System Engineering Concept Demonstration (SECD) Statement of Work", PR B-0-3322, Rome Laboratory, Griffiss Air Force Base NY, 30 August 1989.

"System Engineering Concept Demonstration (SECD) Final Technical Report", Contract Number F30602-90-C-0021, Rome Laboratory, Griffiss Air Force Base NY, draft, May 1991.

AUTHOR'S BIOGRAPHY

Mr. Frank S. LaMonica is a computer scientist at Rome Laboratory. He is the group leader for the Software Environments technology area in the Software Engineering Branch of the Command, Control and Communications (C3) Directorate. He directs research and development activities in software testing and computer-based system/software life cycle environments. He was responsible for the development of several software testing systems, has most recently managed development of the Software Life Cycle Support Environment (SLCSE), and is actively involved with Rome Laboratory's SLCSE Enhancements and Demonstration Program. Current emphasis is in the area of Systems Engineering; in particular, an exploratory and advanced development program whose objectives are to develop and demonstrate state-of-the-art automation for systems engineering.

Mr. LaMonica received a B.S. degree in Electrical Engineering from Clarkson University in 1970 and an M.S. degree in Computer and Information Science from Syracuse University in 1979. He is a member of the IEEE Computer Society and a senior member of the American Institute of Aeronautics and Astronautics (AIAA). He is currently serving on the AIAA Softwar Systems Technical Committee.

CATALYST: AUTOMATING SYSTEMS ENGINEERING IN THE 21ST CENTURY

Edward R. Comer
Software Productivity Solutions
122 Fourth Avenue
Indialantic, FL 32903

Abstract. This paper describes the concept for a future automated environment supporting systems engineering, named Catalyst. Catalyst will enhance the effectiveness of the systems engineering process to pro-.uce more successful, higher quality systems. Sponsored by the Air Force Rome Laboratory, the Systems Engineering Concept Demonstration Program researched the needs and defined the requirements and conceptual design of Catalyst. The paper discusses the operational concept, architecture, and important features and characteristics for Catalyst.

INTRODUCTION

The System Engineering Concept Demonstration (SECD) program was an exploratory effort investigating advanced automation to improve the effectiveness of the systems engineering process. Sponsored by the Air Force Rome Laboratory, the SECD program produced the specifications for an advanced development prototype to be developed in the three to five year time frame. The production version of the envisioned system, termed Catalyst, will provide advanced automated support for systems engineering in the 21st century. The SECD project is described in detail by (LaMonica 1992).

Catalyst will provide an automated environment of integrated, state-of-the-art software tools and methods which supports systems engineering throughout the life cycle. Catalyst will automate a wide variety of existing system engineering processes and provide a user-tailorable framework for evolving and improving the system engineering process.

SCOPE OF SYSTEMS ENGINEERING

Most will agree that systems engineering is vitally important for the successful development and evolution of large, complex, mission-critical systems. But what exactly is systems engineering? Despite the fact that systems engineering is widely practiced, we found it to be a surprisingly nebulous topic.

Systems engineering, as defined by (Blanchard and Fabrycky 1981), commences with the earliest phases of the system life cycle and continues on until the system is retired from use. Systems engineering involves a complex series of activities necessary to:

- transform an operational need into a description of system performance parameters and a preferred system configuration using an iterative process of functional analysis, synthesis, optimization, definition, design, test, and evaluation;
- integrate related technical parameters and assure compatibility of all physical, functional, and program interfaces in a manner that optimizes the total system definition and design; and
- integrate performance, producibility, reliability, maintainability, manageability, supportability, and other specialties into the total engineering effort.

Many of these activities are specific to the type of system being developed. The term "systems engineering" is broadly applied to a wide range of activities that span the scope of whatever is labeled "the system."

We found that systems engineering activities vary significantly between organizations—even between individuals. This variation exists because different organizations do business differently, and systems engineering is the essence of how organizations accomplish their systems business. Even who is labeled a "systems engineer" varies greatly from organization to organization.

Most large system developments employ a multi-disciplinary systems engineering approach that, today, has been labeled as *concurrent engineering*. Concurrent engineering is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to have the systems engineers, from the outset, consider all elements of the product life cycle from conception through retirement, including quality, cost, schedule, and user requirements. (Winner 88) While concurrent engineering is being promoted as a new approach to systems engineering, our analysis of systems engineering indicated that the basic principles of concurrent engineering have been widely practiced for some time.

Early in the SECD program, we had to come to terms with the fact that systems engineering simply means something different to everyone. Questions of what are "systems," "systems engineering," "system design," "system acquisition," or "concurrent engineering" cannot be answered with any general consensus. Rather than languish in the terminology, the effort focused on identifying a set of systems engineering activities that are most popularly associated with systems engineering.

SYSTEMS ENGINEERING ACTIVITIES

A more cohesive view of systems engineering activities emerged after comprehensive study of the applicable literature in systems engineering, interviews with systems engineers, and an extensive process modeling activity (described in (Ortiz 1992)). Systems engineering activities were categorized as follows:

1. **Engineering**, consisting of requirements engineering, system design & allocation, interface definition & integration, trade-off analysis, engineering decision making, change impact analysis & management, integration planning & management, quality engineering & assurance, and specification generation.

2. **Communication**, involving collaboration & coordination, information research, boundary spanning (see (Krasner 1987)), and joint work product development.

3. **Management**, including standards & policy application, process management, program planning & tracking, task management, and risk analysis.

The precise mixture of engineering, communication and management activities varies with each systems engineer and will often change over the life cycle. For example, the systems engineer may begin by performing primarily an engineering role, defining requirements as part of a small team. As the project expands into system design, large amounts of the systems engineers time may be spent coordinating the activities of the subsystem engineers and disseminating a common understanding of the requirements. Once the systems design is complete and the requirements allocated, the systems engineer may be largely performing a management role.

Classified and proprietary information was found to be a recurring complication that impacts all of the systems engineering activities. Classified or proprietary information is often generated and communicated as part of the process. Many DoD systems involve sensitive information to the extent that all or some part of the system is classified. Even systems that are developed in an unclassified setting often have secret threat or mission information and confidential system requirements.

SYSTEMS ENGINEERING TEAM

Today, systems engineering is necessarily a team activity. The growing scale and complexity of mission-critical systems is creating an increasing dependence on collaborative systems engineering employing a number of specialists. In most cases, the systems engineer is supported by numerous other specialty engineers, consisting of subsystem engineers (e.g., hardware, software, electrical, mechanical, structural), analysts (e.g., mission analysis, reliability, human factors, producibility, supportability), and technology specialists (e.g., electronic warfare, sensors, weapons).

A single person can no longer have all of the requisite knowledge and experience in all of the many specialty disciplines to effectively engineer large, complex systems. The systems engineer is a generalist whose knowledge may span many of the disciplines, but must rely on specialists for deep knowledge in single disciplines.

During the life cycle, the systems engineers are the catalyst for the engineering of systems, directing and coordinating numerous specialty subsystem, analysis and technology engineering activities. While the specialty engineering roles often have the ultimate responsibilities for developing or enhancing their respective segments of the system, the systems engineer is unique in his total system responsibility. Because of its broad scope, automating systems engineering is a tall order.

CATALYST

The envisioned Catalyst automation targets the systems engineering team as its users. It focuses on automating the systems engineering process as a group activity. This perspective makes Catalyst effective for modern concurrent engineering approaches. Catalyst seeks to enhance the effectiveness of the systems engineering team, thus producing higher quality systems within cost and schedule constraints.

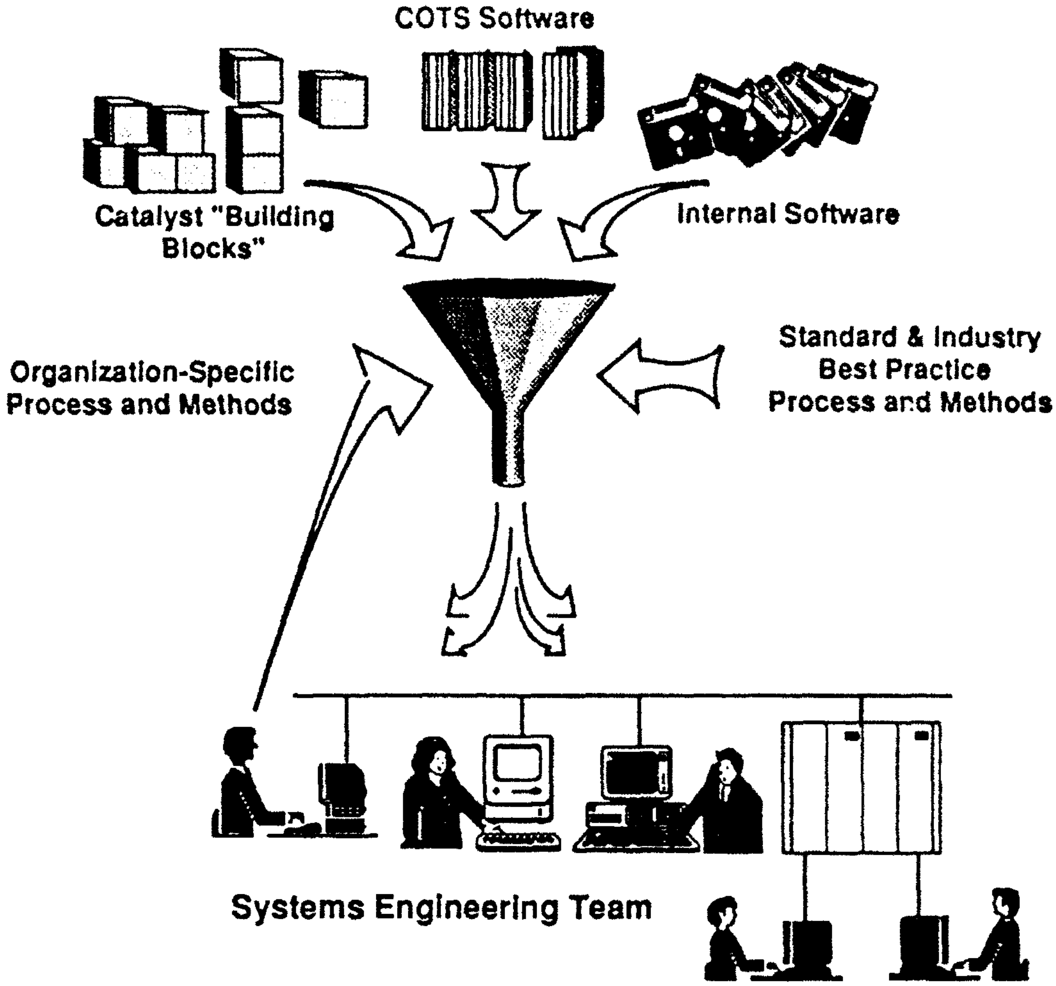

COTS Software
Catalyst "Building Blocks"
Internal Software
Organization-Specific Process and Methods
Standard & Industry Best Practice Process and Methods
Systems Engineering Team

Figure 1 Catalyst Systems Engineering Environment

Catalyst provides automated support for engineering, communication and management activities, as required, depending on the specific organizational, individual or program needs. To allow users to select only those capabilities that are needed, Catalyst is organized as a set of highly adaptable and configurable software "building blocks," as depicted in Figure 1.

The Catalyst building blocks consist of interactive software tools and environment integration mechanisms. When integrated with an installed computer system (hardware plus system software), other software (tools and frameworks), and a systematic process (process models and methods), these building blocks provide comprehensive automated support for systems engineering.

The Catalyst automation focuses on high-payoff, high-leverage automation of mainstream systems engineering activities, integrating a variety of commercial-off-the-shelf (COTS) capabilities and filling automation gaps, where necessary. The areas of Catalyst emphasis are as follows:

- **Process automation**, providing specialized support for defining, planning, tracking, enforcing and improving the systems engineering process.
- **Modeling "meta-tools,"** automating system requirements, design and modeling methods not currently supported by specialty tools, and only partially served by general purpose tools.
- **Concurrent engineering "groupware,"** providing specialized support for the collaborative activities of a multi-disciplinary systems engineering team.

- **Integration mechanisms and specialized tools to exploit integration,** supporting tool interoperability and information management in a distributed, heterogeneous computing environment. Catalyst provides integrated support for information retrieval, documentation, traceability and change impact analysis. Catalyst is designed to be integrated with an installed base of systems engineering, specialty engineering and general purpose tools.
- **Catalyst administration tools,** supporting the systems administrator and security administrator with specialized facilities for installation, integration, administration, extension and monitoring.

The following subsections overview each of these areas and provide our vision of Catalyst, automation for systems engineering in the 21st century.

Process Automation. Catalyst provides process automation to define, monitor, and control the complex systems engineering activities that occur. By being "process knowledgeable," Catalyst assists the systems engineer in managing the system development or maintenance process and provides sophisticated capabilities for process improvement. Catalyst capabilities for process automation include process management, program planning and tracking, task management, standard and policy application, and risk analysis.

Catalyst integrates classic program management capabilities (i.e., PERT process planning, Gantt scheduling, Work Breakdown Structures, resource assignment, cost estimation, and tracking) with comprehensive process and task management facilities. Program plans are generated that are guaranteed to comply with applicable standards and policies. Tasks identified in program plans can be directly assigned, executed, tracked and enforced. Interfaces are provided to external organizational accounting and reporting facilities.

Catalyst provides facilities to define process models, organizations, roles, policies and procedures. Process definitions are used to guide the process defined by the systems engineer or program manager. Process alternatives may be planned and then later enabled to respond to specific risks. Catalyst supports automated enforcement for process and product standards and policies. Catalyst's process automation support reduces the amount of human communication and monitoring that is required to execute plans.

For example, process automation capabilities may be used to automatically support and enforce the dissemination, routing, review and sign-off of contractual deliverables or important decisions. Automated monitoring mechanisms are provided to allow automated "watchdogs" to be enabled to monitor the process and/or products for significant events or thresholds.

Catalyst supports the instrumentation of the systems engineering process to provide meaningful and up-to-date measures of progress and efficiency. Catalyst supports many different visual presentations to assist in metrics interpretation. A full range of statistical analysis capabilities are provided to support statistical quality control.

Modeling Meta-Tools. Modeling is widely applied for requirements engineering, systems design and allocation, interface definition and tradeoff analysis. Systems engineers employ a variety of informal, semi-formal and formal modeling methods throughout the life cycle.

Most available automation is either general purpose tools (e.g., word processors and graphic drawing tools) or special purpose tools supporting more formalized methods (e.g., simulation or CASE tools). Catalyst will target the middle ground, as illustrated in Figure 2, automating the semi-formal methods that may reflect industry best practices or may be unique to a particular organization or even individual. To support the informal and formal ends of the spectrum, Catalyst will provide interfaces to the user's general purpose or specialized modeling tools.

Catalyst's meta-tool capability allows the systems engineer to quickly create an automated capability for modeling techniques that are currently manual or inadequately supported by automation. New modeling tools may be created by the user with an intuitive "select and drag" user interface. The approach provides automated support for an organization's existing modeling methods and empowers the organization to incrementally change and improve its automated systems engineering process.

New modeling methods are defined by determining the desired presentation symbology and syntax, the information content, consistency and completeness rules and diagnostics. Several different types of representations are supported, including text forms, textual languages, line graphics, tables, equations, outlines, hierarchies, plots and graphs. A user-defined modeling method may include many different model

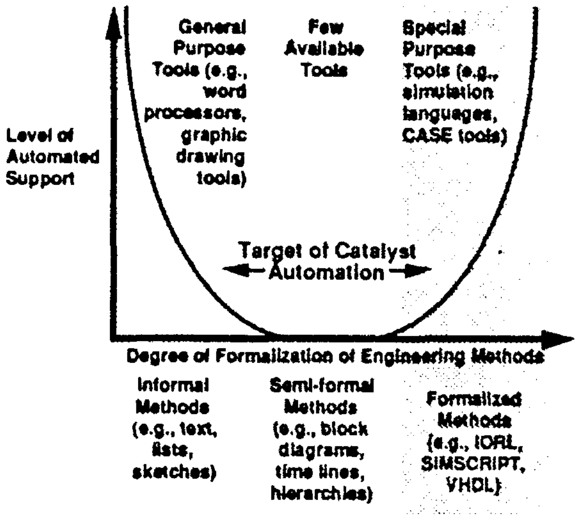

General Purpose Tools (e.g., word processors, graphic drawing tools)
Few Available Tools
Special Purpose Tools (e.g., simulation languages, CASE tools)
Level of Automated Support
Target of Catalyst Automation
Degree of Formalization of Engineering Methods
Informal Methods (e.g., text, lists, sketches)
Semi-formal Methods (e.g., block diagrams, time lines, hierarchies)
Formalized Methods (e.g., IORL, SIMSCRIPT, VHDL)

Figure 2 Catalyst Meta-Tool Applicability

views and may be structured into multiple levels of abstraction. User defined semantic rules maintain consistency between views and levels.

For example, this capability supports graphic models such as time line models (for example, as described by (Pariseau and Stuebing 1992)), process models, enterprise models, system block diagrams, and data flow models. Not only can a user draw the associated diagram, but the content of the diagram (e.g., the functions defined, or the interfaces identified) is automatically stored in the project database for use and reference elsewhere. Because the system supports models that provide alternative views of the same information, changing one representation automatically updates all other views of the same information to be consistent. These multiple views better support the understanding and review of system requirements by the customer, user and specialty engineers.

Catalyst allows the user to import previously informal text or graphics and to incrementally formalize it. For example, a system block diagram that defines the subsystems and interfaces is originally done by hand. The graphic is scanned in and identified as a previously defined type of model representation of compatible symbology. Catalyst not only is able to display this graphic electronically, but also supports its interactive editing, and automatically creates in its database the subsystems and interfaces identified in the model.

Concurrent Engineering Groupware. All of Catalyst's capabilities are fine tuned to support the various collaborative and team activities that naturally occur in a concurrent engineering process. Catalyst's "groupware" automation allows most tasks, accomplished today in a singular fashion, to be accomplished by teams that are working in a parallel, but coordinated, manner. Group coordination support is provided for group processes such review, discussion, approvals, meetings, information dissemination, information requests, polling, and brainstorming. Specialized group processes, such as issue analysis, interface definition & management, and tradeoff analysis, are also supported.

Catalyst addresses work product development as a group activity. Individuals employ document templates for consistent and compliant work products. Documents may be partitioned and assigned to different persons. Simultaneous work product editing by multiple users is supported. Using an annotation facility, users may comment on and critique work products in process.

Facilities are provided to interface with specialty engineering environments to support multi-disciplinary activities such as tradeoff analysis. For example, results from multi-disciplinary analyses accomplished by several individuals using various external tools are able to be coalesced to support decision making. Catalyst records the various analysis alternatives, the results of the analysis, the decision and the rationale for later reference. Mechanisms are provided to track the decision to ensure that it has been executed properly. By maintaining traceability to those areas of the system definition that were involved in the trade-off and maintaining the relationship between trade-offs, reviewing or revisiting trade-offs is supported.

Catalyst naturally supports modern electronic communication, including electronic mail and electronic bulletin boards. Any form of electronic information may be routed or shared. Electronic notification of key events (e.g., establishing a new baseline) is supported. The electronic communications interoperate with other systems through standard interfaces to allow communication across organizational or geographic boundaries.

Integration Mechanisms and Specialized Tools to Exploit Integration. Catalyst provides integration mechanisms for tool interoperability and information sharing in a distributed, heterogeneous environment. These facilities integrate the various Catalyst building blocks and integrate Catalyst with external systems engineering, speciality engineering or general purpose tools.

Catalyst's integration mechanisms are based upon a set of emerging object-oriented industry standards. A highly flexible object messaging backbone allows tools to communicate information and events between tools and to invoke and request services of other tools. Catalyst provides object management facilities to serve as a repository for project and system information.

Catalyst is designed to be integrated into the user's existing computing environment. Catalyst is interoperable with a user's existing software tool suite. Users may extend Catalyst by integrating other user tools (COTS or custom) into the environment, including general purpose tools (e.g., editors, drawing packages, spreadsheets); external information databases, object bases or infrastructures; project management software; metrics tools; modeling and simulation tools; analysis and display software; document formatting, production and publishing systems; and configuration management software. Catalyst supports Computer-Aided Acquisition and Logistics Support (CALS) and industry-standard data interchange formats.

The object messaging backbone is used to integrate a collection of Catalyst building blocks and other tools to allow exploitation of all of the information and relationships across the environment without requiring that all information be contained in a single, central repository. As a result, the Catalyst user has a variety of powerful capabilities that exploit the connectivity and integration, to accomplish information retrieval, display and editing; allocation and traceability; change impact analysis and management; and document generation.

To support handling of classified information, Catalyst supports and uses the underlying capabilities of a Trusted Computing Base certified up to the B2 level (according to DoD 5200.28-STD DoD Trusted Computer Systems Evaluation Criteria). The underlying Trusted Computing Base employs sufficient hardware and software integrity measures to allow its use for restricted processing of classified and/or sensitive information.

Catalyst Administration Tools. Catalyst supports the systems administrator and security administrator with specialized facilities for installation, integration, administration, extension and monitoring. Our experience has shown that integrating and maintaining a large, diverse toolset, such as Catalyst, requires automated support.

Catalyst supports the definition of user roles and the assignment of environment and information access privileges according to individual user or role. The system supports the collection and reporting of usage metrics, audit trails and current environment status. A comprehensive set of environment self-test and diagnostics are included.

Catalyst provides specialized tools for the security administrator to define, implement and monitor discretionary and mandatory security policies. Specific capabilities include support for conducting security risk analysis on the configuration and policies; defining, implementing, and enforcing security policies; monitoring the environment for security-relevant events and investigating potential security violations; and supporting trusted configuration management of critical Catalyst building blocks.

CATALYST FLEXIBILITY

Catalyst will be extremely flexible to accommodate a wide spectrum of systems engineering approaches and needs. Catalyst will empower its users to change and evolve its automated features to incrementally improve the systems engineering process.

Catalyst will be adaptable to accommodate variability in process and life cycle; organization structure and roles; methods and techniques; information and representations; work flows and work products; policies and procedures. Catalyst will be portable and configurable to a variety of computing platforms and networks. Catalyst will be interoperable with industry standard interfaces and will be designed to integrate with the installed base of general purpose and special purpose automated tools.

Flexibility will be the key to Catalyst's success. Catalyst will automate the way an organization does systems engineering and change as their process evolves. Catalyst's flexibility will speed the transfer of this exciting technology to automate the high payoff systems engineering field.

REFERENCES

Blanchard, Benjamin S, and Wolter J. Fabrycky, *Systems Engineering and Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

DoD 5200.28-STD, *DoD Trusted Computer Systems Evaluation Criteria.*

Krasner, Herb, et. al., "Communication Breakdowns and Boundary Spanning Activities on Large Programming Projects," *Empirical Studies of the Design Process: Papers for the Second Workshop on Empirical Studies of Programmers*, MCC Technical Report Number STP-260-87, September 24, 1987.

LaMonica, F., "System Engineering Concept Demonstration - An Overview," *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering (NCOSE)*, Seattle, WA, 20-22 July 1992.

Object Management Group, *Common Object Request Broker (CORBA), Architecture and Specification*, 1992.

Ortiz, A., "System Engineering Concept Demonstration (SECD) Process Model," *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering (NCOSE)*, Seattle, WA, 20-22 July 1992.

Pariseau, Richard J. and Stuebing, Henry G., "System Engineering of Naval Air ASW Systems,"*Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering (NCOSE)*, Seattle, WA, 20-22 July 1992.

Redwine, Samuel T., et al., *DoD Related Software Technology Requirements, Practices, and Prospects for the Future*, Institute for Defense Analysis, Prepared for the Office of the Under Secretary of Defense for Research and Engineering, IDA Paper P-1788, June 1984.

Winner, Robert I., et al. *The Role of Concurrent Engineering in Weapon Systems Acquisition.*, Institute for Defense Analysis, prepared for the Assistant Secretary of Defense for Production and Logistics, IDA report R-338, December 1988.

AUTHOR'S BIOGRAPHY

Edward R. Comer is President of Software Productivity Solutions, Inc., involved in the development of advanced software and systems engineering methods and tools.

Mr. Comer has over 15 years experience in developing, modeling and refining engineering methods for large, mission-critical computer systems. His systems engineering experience spans numerous application areas including avionics, real-time process control, intelligence data handling, digital signal processing, communications, interactive data management, composition and word processing systems.

Mr. Comer is a member of the National Council on Systems Engineering (NCOSE), the American Institute of Aeronautics and Astronautics (AIAA) Software Systems Technical Committee, and the Institute of Electrical and Electronics Engineers (IEEE). Mr. Comer has published numerous papers on system modeling and analysis and on advanced system and software engineering methods.

He holds a B.S. degree in Mathematics and M.S. degrees in Computer Science and in Applied Mathematics from the Florida Institute of Technology.

SYSTEM ENGINEERING CONCEPT DEMONSTRATION (SECD) - PROCESS MODEL

Alberto Ortiz
McDonnell Douglas Corporation
Douglas Aircraft Company
3855 Lakewood Blvd.
Long Beach, CA 90846

Abstract. This technical paper will present a summary of the SECD Process Model Task sponsored by Rome Laboratory at Griffiss Air Force Base, New York. An introduction to the SECD Program is presented in the "*SECD - Overview*" [LaMonica 1992]. The SECD Process Model is a system acquisition and development model with emphasis on system engineering activities over the entire system life cycle. The Process Model is a graphical and textual representation of the system engineering life cycle activities, agents, flows, feedbacks, and work products. This interactive Process Model provides a multi-dimensional view of government acquisition and contractor development activities. In the context of the SECD Program, the Process Model validates the System /Segment Specification (SSS) requirements and demonstrates coverage and completeness of the system engineering process.

BACKGROUND

By emphasizing system engineering activities, the Process Model allows us to better represent and customize these activities within their natural domain. The Process Model includes a list of standards in order of precedence to provide validity and traceability to commonly used and approved sources. For the sake of completeness, the processes captured in the model are based on formal and informal activities not previously captured or formalized by these types of models.

In this paper, we will present details, figures and descriptions in the development of the Process Model. Future consideration will be given to the implementation of system metrics and algorithms to measure maturity and fidelity of a system as part of the model. Topics that will be discussed include requirements overview, acceptability criteria, strategy and approach, precedence list of standards, high level portrait, discriminating factors, metrics and instrumentation, lessons learned, and future plans-applications and directions.

System Engineering Concept Demonstration (SECD) is an exploratory development effort sponsored and managed by Rome Laboratory to specify a system engineering environment which came to be known as "*Catalyst*". The SECD contract was awarded in February 1990 to Software Productivity Solutions (SPS) Inc., Indialantic FL, as the prime contractor. McDonnell Douglas Corporation - Douglas Aircraft Company (MDC-DAC), Long Beach CA, and MTM Engineering Inc., McLean VA, are subcontractors [LaMonica 1992].

McDonnell Douglas Corporation-Douglas Aircraft Company's (MDC-DAC) role and task in the SECD program is to provide SPS, Inc. and Rome Laboratory insight and advice on system engineering processes, policies, and practices from a large prime contractor's point of view, and to develop a system engineering life cycle Process Model. Our participation ensured a strong system perspective in the development of the "*Catalyst*" environment. As we move into the 21st century, MDC DAC is committed to improve the quality, cost and performance of our systems. Our interest in "*Catalyst*" resides in the fact that system engineering and the automation of the system engineering process is a key to a more competitive, higher quality, and improved performance of our product line. In the following sections, we present a summary of the Process Model development process, approach, effort, and its inception.

REQUIREMENTS OVERVIEW

Goals. The Process Model goals and objectives are as follows:

1) Verify "*Catalyst*" requirements to demonstrate coverage of the system engineering activities by analysis.
2) Prepare an example of a System Engineering Process Model (SEPM) for use as an environment training tool.
3) Identify the default system engineering processes for initialization of "*Catalyst*".
4) Provide general insight into system engineering activities, interfaces, work products,

techniques, etc. to SPS, Inc. and Rome Laboratory.
5) Ensure a high utilitarian value and usability of "*Catalyst*" as an end product.

Acceptability criteria and goals supporting steps were so developed to provide an overall direction to the 'fort. These criteria and steps were needed to achieve ɹr goals and objectives.

;oals Supporting Steps. The Process Model task ɔals supporting steps are the road map to the effort. hese steps convey the path followed in the evelopment of the Process Model. The goals ɹpporting steps are technical library searches, ocumentat:on reviews, program field interviews, data ıodel views, and abstract model views. Descriptions of ıe aforementioned steps are available in the *"Process fodel" Final Technical Report (FTR).*

)bjectives. The objectives of the Process Model task re to identify the system engineering process during the ntire system life cycle, demonstrate coverage of the ystem engineering process by the *"Catalyst"* nvironment requirements, and to adapt the Process 1odel representation to the SECD Prototypes and)emonstrations Scenarios.

The conceptual requirements for the Process Model re a good understanding of the system engineering rocess, ensure completeness of *"Catalyst"* equirements, establish the basis for the Operational :oncept Document (OCD), provide an infrastructure for ıe prototype scenarios, and incorporate the results into ıe prototypes and demonstrations. These conceptual equirements for the Process Model are based on ustomer needs and they are the basis for establishing an cceptability criteria.

ACCEPTABILITY CRITERIA

The acceptability criteria of a system or work product Process Model) is relative to its utility in relation to a et of customer value standards [Chestnut 1965]. The :ustomer must assess a value judgement about the ystem or work product. Typical customer value tandards include performance, cost, reliability, time, ınd maintainability. An acceptable system or work ɹroduct may or may not meet specified requirements. In he case of the Process Model, the acceptability criteria ıugmented the requirements in the SOW, goals, and ɔbjectives. The Process Model acceptability criteria 'actors are readability, traceability, acceptability, ɹniformity, usability, changeability, consistency, and nteroperability. These criteria factors were developed in ın evolutionary manner through experimentation with ⁄arious representations, methods, and tools. The ıcceptability criteria was the result of intense customer interaction and involvement with several representation ideas. Explicit descriptions of the acceptability criteria factors are available in the *"Process Model" Final Technical Report (FTR)* [Ortiz 1992].

STRATEGY AND APPROACH

Our modeling strategy and approach was to meet the customer requirements, needs and acceptability criteria by experimentation. These experimentations included IDEF0, IDEF1, Delta Charts, MacDraw, Embedded Computer System Analysis Method (ECSAM), and others. More details on the experiments conducted are available in the *"Process Model" Final Technical Report (FTR)* and *"Process Model" Supporting Document* [Ortiz 1992].

Our experimentations unveiled the issue of process variation. The system engineering process not only varies throughout the system life cycle, but from organizat:on to organization, within an organization, and from person to person. The challenge in process variation is ascertaining how to develop a representation approach that supports various system engineering methods, processes and practices. Our selected approach was to represent activities and processes in a *"generic"* manner. This approach will allow system engineering practitioners to recognize and tailor the model. This is one of the reasons for a *"generic"* Process Model.

Figure 1. illustrates the vertical and horizontal variations of the system engineering process within the system life cycle. These variations are the result of conflicting source documents and standards as well as, differing organizational practices and roles, types of systems, and personal preferences.

Another experimentation issue was conflicting source documents and standards. We observed conflicting directives, even within the same standard. To resolve this conflict, we developeu a list of standards and provided in o-der of precedence. This list of standards allows us to prioritize directives, standardize name labels, and validate activities at higher levels of abstraction. This is the foundation of the Process Model traceability and usability characteristics. Since variations occur across the board, acquisition and engineering standards were interlaced together to portray the overall life cycle process. This proved to be close to real life practices.

The system engineering process variations appeared again during our program field interviews. Not only was the system engineering process emphasized differently in other organizations, but it also varied within the same organization. Our Program Field Interviews were held at the Naval Air Warfare Center

Aircraft Division-Warminster, PA, Rome Laboratory-Rome, NY, IBM-Owego, NY and MDC-DAC-Long Beach, CA. These field interviews revealed that the system engineering discipline is practiced at the acquisition, development, and sub-system engineering levels.

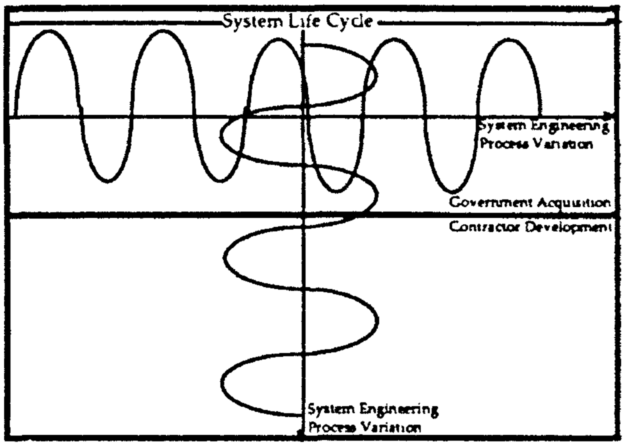


Figure 1. System Engineering Process Variation

System engineers at Rome Laboratory emphasized studies, communications, and program management while practitioners at Naval Air Warfare Center Aircraft Division-Warminister emphasized operational needs and specialty engineering practices. At IBM, the emphasis was the system engineering process management, and at MDC-DAC, the emphasis was program and supplier management and specialty engineering. The system engineering process variations among the organizations were diverse.. Even in the program management area from government to contractor, we found variations in the system engineering process. To satisfy this broad base of differences, the Process Model represents government acquisition and contractor development with emphasis in system engineering activities. The Process Model supports the three basic groups of the system engineering roles which are engineering, management, and communication [Comer 1992].

Our approach used the acceptability criteria, previously introduced, to develop a representation methodology and select a tool that is consistent with this criteria. We chose *"MacFlow"* an Apple Macintosh based flow charting tool because it supports the representation of ANSI standard symbols. These symbols allow us to better understand and read the Process Model. In addition, *"MacFlow"* supports the hierarchical representation of a process in various interactive modes. This functionality allows us to abstract complex processes and interactions into simplified representations. Readability was the main discriminating factor with the majority of the representations and tools researched. Table 1 describes the Process Model symbology and color codes.

Symbol	Description	Color Code
Parallelogram	Input/Output	Turquoise
Rectangle	Process	Light Purple
Diamond	Decision	Yellow
Circle	Review	Green
Tombstone	Milestone	Hot Pink
Hashed Rectangle	Process Grouping	Light Purple (Perimeter Only)
Cross Hashed Rectangle	Input/Output Grouping	Turquoise (Perimeter Only)
Line	Flow	Blue
Hashed Line	Feedback	Dark Red

Table 1. Process Model Symbology Description

Our strategy consists of using customer value standards to select and refine the Process Model representation. This strategy pays off in the final deliverable product.

PRECEDENCE LIST OF STANDARDS

During our technical library searches thousands of source documents and standards were unveiled and reviewed. The need arose to organize and manage this information effectively. The precedence list of standards allows us to prioritize, validate and trace activities and processes into commonly used and approved sources. In turn these sources are used to identify activities and processes within the system life cycle. Numerous conflicts are resolved between standards, directive. and documents using the precedence list of standards. The precedence of these standards was determined by the issuing agency (i.e. DoD, Air Force, Navy, Army, etc.), relevance of information with regards to the model, and granularity or detail of the information contained within. No preference was given to any service or type of standard considered. The precedence list of standards, in order of precedence, are DoDI 5000.2, MIL-STD-1521B, MIL-STD-973 (Not Approved), MIL-STD)-499A and B, DOD-STD-2167A, AFR 800-14, and S&E Instruction 5421.2 (Naval Air Warfare Center).

HIGH LEVEL PORTRAIT

The Process Model has eight levels of detail. To describe the model, it is necessary to provide a High Level Process Model Portrait. This portrait describes the organization, abstraction, and assumptions of the

Process Model. We emphasize unique areas of the process in which assumptions about the system engineering process were made. This is an early overview of the Process Model.

Organization. The Process Model is organized in hierarchical levels of abstraction. Each level expands the previous level of abstraction. The Process Model is integrated to allow this type of navigation interactively. Figure 2. Process Model Organization, illustrates the upper three levels of the Process Model.

Level 1. The first level of the Process Model is called *"System Acquisition And Development Process Model With Emphasis On System Engineering Activities"*. It represents the system life cycle activities for each phase of system development. The phases and boundaries are represented with process and the milestone symbols respectively. Milestones are place markers between the system phases represented as processes. These symbols are enclosed by a process grouping symbol. This enclosed organization is one of the major assumptions of the Process Model task. Although the real life organization could not be enclosed, this assumption was necessary to scope the task and establish the domain boundaries.

Level 2. The second level of the Process Model is called *"System Acquisition and Development Top Level Process Model with Emphasis in System Engineering Activities"*. It expands the first level and sub-divides the system development phase between government acquisition and contractor development. At this level, the milestone symbols can be seen inside each phase denoting boundaries between system development phases.

Level 3. The third level of the Process Model is called by each individual phase in accordance with DoDI 5000.2. At this level of abstraction lower level processes and input/output symbols can be seen. The separation between Government Acquisition and Contractor Development is maintained. Now, the three system engineering process groups are visible at this level. These groups are requirements, analysis and planning. Another eight levels of varied detail are available through most individual process symbols.

Assumptions. Many assumptions were made in order to develop the Process Model. Within the context of this section, Assumptions will refer to specific representations in the Process Model. Two major assumptions were chosen for discussion. These assumptions relate to the representation of the system engineering process within the Engineering & Manufacturing Development Phase and the Production & Deployment and Operations & Support Phases. Figure 3. and 4. illustrate these assumptions.

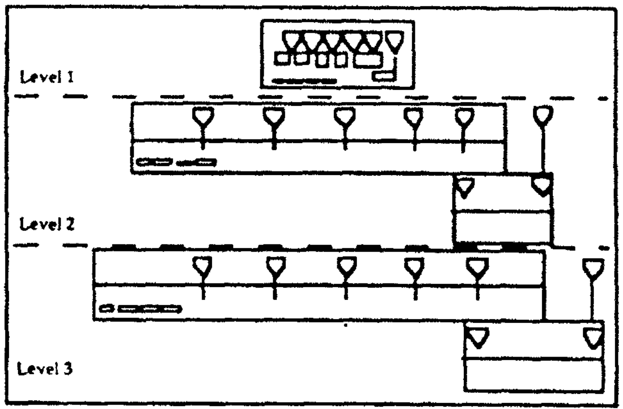


Figure 2. Process Model Organization

1) The first assumption is the separation of the hardware, software and system engineering processes. We assumed that hardware, software, and system integration are specialty areas and not the thrust of the system engineering process. Figure 3., Hardware, Software, and System Engineering Process, shows how the system engineering process is fed backwards by the hardware and software development processes. Since the output of the system engineering process defines the system, each formal hardware and software development review feeds a greater level of detail to the system engineering process which in turn ensures consistency and refines the system definition work products. Worth noticing is the possibility of using three different development approaches (i.e. spiral, incremental, concurrent, etc.) for the hardware, software, and system engineering development. The Integration and Test process puts it all together. In real life, Integration and Test specialists uncover numerous design, performance, and compatibility problems. The Integration and Test process is key to a deliverable system. The system is usually integrated incrementally.

2) The second assumption is the connection between the Production & Deployment and the Operations & Support phases (Figure 4). The milestone connection between the previously mentioned phases is *" Major Modification Approval"*. As we enter the Production & Deployment Phase the Government must assess the requests it receives from the field in order to establish operational effectiveness. Figure 4. illustrates this point by showing an output from the Government assessment process in the Operations & Support Phase and the Production and Deployment Phase.
Notice that the system first article output is aligned with the *"Major Modification Approval"* milestone.

This means that it is possible to deploy a system straight from the production line in which case field operations and support must provide feedback for any needed modification of the system. As the production line continues, changes to the system can be implemented. The Process Model accounts for this possibility in its representation. Any changes to the system must be formally qualified. This is the reason for the FCA, PCA, and FQR reviews at the later part of the Production & Deployment phase. The Mission Needs & Requirements output contains the Statement Of Need (SON) used at the beginning of the system life cycle. The last milestone (left to right) in Figure 4. is *"System Retirement"* at which point the SON identifies the need for a suitable replacement.

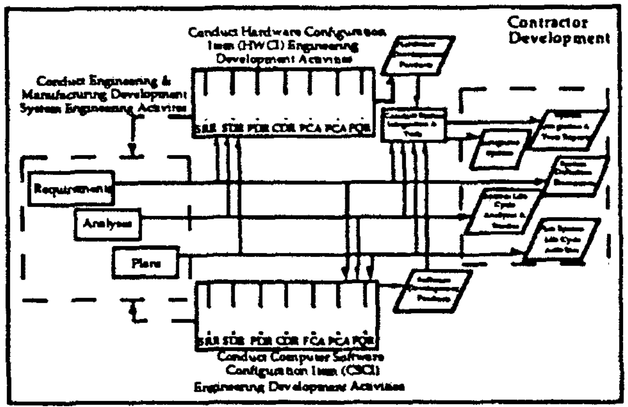

Contractor Development
Conduct Hardware Configuration Item (HWCI) Engineering Development Activities
Conduct Engineering & Manufacturing Development System Engineering Activities
SRR SDR PDR CDR FCA PCA FQR
Requirements
Analyses
Plans
SRR SDR PDR CDR FCA PCA FQR
Conduct Computer Software Configuration Item (CSCI) Engineering Development Activities

Figure 3. Hardware, Software, and System Engineering Process

DISCRIMINATING FACTORS

The Process Model has a number of discriminating factors. Following is a list:

a) Emphasis in communicating information about the process.
b) The interactions between Government Acquisition and Contractor Development are captured.
c) The eight hierarchical levels of detail are represented.
d) The interactive multi-dimensional view of the processes is represented.
e) The system life cycle perspectives are modeled including the system engineering process.
f) A generic representation was selected
g) The integration of various system engineering standards into a cohesive and understandable view.

The Process Model is not the final solution. The objective of any model is to streamline the process. We encourage further experimentation with the Process Model. This Process Model has not been implemented or tested in a real program yet. Many of the aforementioned discriminating factors differentiate our works from others, but the main one is that it provides essential information about the process.

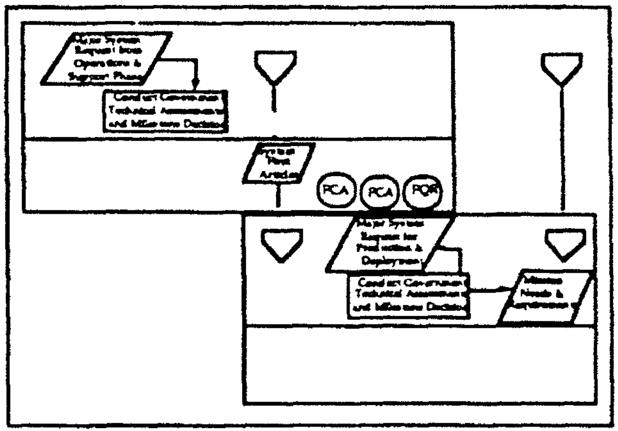


Figure 4. Process Model Production & Deployment and Operation & Support Phases

METRICS AND INSTRUMENTATION

In this section we discuss the instrumentation and application of metrics to the Process Model. The writing was provided by Mr. Mike Carrio, MTM Engineering Inc., McLean VA.

The Process Model enables the collection of predictive metrics in a cohesive and consistent manner. Predictive metrics, unlike their traditional counterpart, post-facto metrics (e.g. complexity metrics), can now be employed effectively in the early development and acquisition phases to provide risk mitigation strategies and approaches. In the past, the lack of viable process models has precluded early life cycle instrumentation employing an extensive set of predictive metrics.

Additionally, the Process Model provides a contextual framework for the association and mapping of development and design methodologies. Use of formal and defacto methodologies can now be mapped to corresponding processes and work products to assess their breadth and completeness. This mapping in turn provides the basis for quantification of product, usage, and process metrics. Furthermore, relationships between process and product metrics can be algorithmically established to provide greater synergism with the predictive metric domain.

The Process Model, with its capacity for instrumentation can be utilized to effectively establish requirement state vectors with the inherent ability to determine requirements and process maturity, stability and completeness. Where used, formal methodologies (i.e. methodologies supported by a formal grammar or executable notation) can be closely aligned to those processes or activities they are supportive of. The formal notation can then be utilized as an instrumentation vehicle to collect particular mesurand information. The mesurand information in turn can be incorporated into metrics for subsequent quantification. This is identical to using a software implementation language as the vehicle for collecting metrics. Thus, metrics collection can be extended from the software implementation process, well into the systems engineering process and beyond. Quantification is thus enabled much earlier in the acquisition/development phases than previously to allow cost effective implementaticn trade-offs.

One of the barriers to enabling predictive metrics and viable risk mitigation strategies early in the development has been the inability to instrument and collect this type of information due to its informal, undisciplined and undefined nature. While requirements state vectors consists of cost, schedule, and requirements maturity and stability dimensions; and consists of complex algorithms, they lend themselves to easily depicted and simplified representations. These representations can be effectively mapped into program management tools to manage complex systems during their embryonic, but critical activities to provide early warning alarms required by management. Program managers and their staffs, while using the same underlying technical units of measurement as those of the systems and software engineers, are viewing the same issues as their team members, but from their respective frame of reference (i.e. from a cost and schedule view, not requiring knowledge of the state vector coefficients or algorithmic components). This concept is somewhat analogous to the Parnas' concept of information hiding where software package functionality can be separated from the details of package implementation to enable effective communication between individuals at different levels of abstraction. The latter concept is one of the maturing signs of software engineering as a science, providing synergism to the realm of system engineering.

LESSONS LEARNED

The first general lesson is that process modeling is a very complex and mentally intensive task. Do not try to do this by yourself. A team of three to five people should be able to carry on intelligent and organized discussion about any model development issue. A considerable amount of effort is required to model any process and one should not underestimate the task. The weaving together of numerous elements into a cohesive whole is not simple. Ensure ample time is taken for planning the task.

The second general lesson is that nobody really understands the process completely and you are in the process of becoming the expert on it. Everyone knows something about the process and the more you communicate with others; the more effective your modeling skills will become. Knowledge is power.

The Process Model Task lessons learned provide a general insight to the challenges of process modeling. These lessons are as follows:

1) Goals and objective are great, but a detail plan of action consistent with your customer needs is a must in any process modeling task

2) Focus on information gathering, representation method, automated tool selection, and desired level of abstraction.

3) Abstraction and scoping are the key issues to process modeling. They influence the representation form and depth of detail required. Continuously assess abstraction and scope to ensure proper direction.

4) Concentrate in experimentation with various representation approaches. It reduces risk and will help you find solutions to trouble areas.

5) Develop or chose your representation method and automated tool early on. Remember modeling is a discovery process and you do not need to discover that you are lost.

6) Set the domain boundaries to the problem and do not be afraid to make assumptions. A wrong assumption is better than no assumption because it starts the discovery process.

7) Reconcile conflicting information by setting a precedence list. This is very important because it solve ambiguity issues.

8) The objective of any process model is to streamline the process.

9) The model is never done. Processes are like standards, living documents, always adapting to new technologies, practices, businesses, and design constrains. Set a measure of completeness and success as early as possible.

10) Involve your customer in the review of the process model. As they gain understanding and appreciation for the work, they will be supportive of further experimentation.

11) Finally, don't be discouraged and don't be afraid to fail. Most of the process modeling work is original because every model developed has the unique characteristics of its modeler and customer.

The aforementioned lessons were challenges in the Process Model Task. As we look to the future, other lessons will be learned and we will be ready for these challenges.

FUTURE PLANS-APPLICATIONS AND DIRECTIONS

In this section, we outline the future potential for the process model. These include the addition of metrics and instrumentation, the mapping of system methodologies into the model and the utilization of the model as a seamless interface for other works in the system engineering area. We discuss the use of the model as a process knowledgeable engine and the addition of more levels of abstraction to produce system engineering procedures.

Areas of Process Model application as part of *"Catalyst"* include training, tracking, tracing, planning, guiding, etc. The utilization of the model as a reference to benchmark system engineering methodologies is detailed in this section.

We envision the Process Model as a spin off platform for a myriad of applications and experiments in the areas of metrics, environments, development methods, reference model, requirements tracking, training, validation, management, knowledge based and others. Since we can't address everything, we have chosen the following applications to be representative.

Reference Model. It is envisioned that the Process Model, together with such other models as the Software Engineering Institute (SEI's) Capability Maturity Model and the recently released (Jan. 92) National Institute Of Standards & Technology (NIST) Reference Model for Frameworks of Software Engineering Environments represents, for the first time, the ability to bring insight and discipline to processes, methodologies, and environments as well as the services the latter provide. More details about the application of the Process Model as a Reference Model are discussed in the metrics and instrumentation section of this paper.

Process Management And Guide. The Process Model because of its breadth and detail can conceivably be applied to an existing program. It should provide the foundation for identification and management of the numerous activities in a particular phase of the life cycle. By helping plan and direct activities in a program, the Process Model is effectively being used as a management tool. Since the Process Model is based on applicable standards, it becomes a graphical representation of various integrated textu' standards. In the 21st century, the Process Model would be defined visually to the procedure level. One will be able to observe the system parts pulling together as the development of the system is being completed. Cost, schedule, and requirements state vectors will be integrated to provide customized statistical information about the system and its process. *"Catalyst"* will be the basis for the framework of this application.

Training Tool. The Process Model has th potential to become the first system engineering se taught tool. We envision a multi-functional touch an pen screen color display with multi-media capability i which the model provides sample system engineerin tutorials to a student at different levels of experienc Integrated processes and work products can be used t test and ascertain changes to the system developme process. This is a high promise area since the Proce Model works interactively already. *"Catalyst"* woul expand its support base to a training role.

Process Knowledgeable. As *"Catalyst"* require more knowledge of the system engineering process, w envisioned the Process Model as providing the basis f the development of enactable process knowledgeabl based rules that will support varied levels of the syste engineering function. In order to enact the proce knowledgeable rules, we foresee a parametric engine o the system engineering process throughout the lif cycle. This engine will be based on a state vect approach. Experimentation at a small scale in this are is required to determine feasibility of concept.

Metrics And Instrumentation. Metrics an instrumentation is a near term and highly promisin application area of the Process Model. As discusse earlier in this paper, it is possible to instrument th Process Model with predictive metrics and measur these parameters in a mathematical form (See th Metrics and Instrumentation section of this paper f more details). In order for these metrics to be applie effectively, the Process Model needs to be abstracte downwards to develop a set of generic syste engineering procedures. This is a near term task for th next phase of the SECD program. In this phase w envision the integration of the Process Model int *"Catalyst"* as part of a prototype system.

CONCLUSIONS

The SECD Process Model Task was arduous an demanding in all facets. It is the author's opinion th we have broken new ground in the understanding of th system engineering process. This interactive syste life cycle Process Model is extremely comprehensiv flexible, and extensive. It is intended to support th necessary degrees of freedom needed to accommodate th many identified life cycles standards, as well as, new an emerging ones. Its strength lays in the identifie interfaces between government acquisition an contractor development and the interaction between th system engineering process and the system life cycl activities. The following are our conclusions about th Process Model, system engineering, and the syste engineering process.

1) The system engineering process is contained within the system development process which in turn is contained within the system acquisition process.
2) The system engineering process cannot be separated from the system development process. The interfaces between system acquisition and system development must identified for the system engineering process to be meaningful.
3) The system engineering process varies from one organization to another. Its variation encompasses the entire system life cycle horizontally and vertically. Variation of the system engineering process is a key issue in the Process Model Task.
4) The system engineering process can be divided into three groups; requirements, analysis, and planning.
5) The system engineering role can be divided into three groups; engineering, management, and communications.
6) Automation of the system engineering process can be achieved through flexibility and adaptability.
7) System engineers develop models that are represented in various forms such as textual, graphics, scripted, and other notation languages.
8) System engineering processes and methods are derived from accepted industry standards and include best practices for implementation. The Process Model is the means by which the users are empowered to tailor and improve system processes and methods.
9) The Process Model must be generic in order to support the variation and abstraction needs of the Process Model.
10) The Process Model emphasis is the encapsulated information and not the representation. The acceptability criteria ensures that this emphasis is maintained throughout the representation.
11) The Process Model must be instrumented to incorporate system metrics.
12) System Engineering includes Concurrent Engineering (CE), Reverse-Engineering, Re-Engineering, Requirements Engineering, Integrated Process Development (IPD), Design To Life Cycle Costs (DTLCC), Technical Program Management, System Architecture, System Analysis, and Commercial Systems. System engineering does not include hardware or software development or system integration. These disciplines are considered specialities within the Process Model and they have a big impact in the system engineering process and its work products.
13) System Engineering is a problem solving technique and it is applied at all levels of system development, acquisition, and management.
14) The Process Model by virtue of its extensibility and completeness can be used to customize systems engineering processes unique to each individual development approach.
15) The Process Model can be used as a Reference Model to map system engineering methods and assess their completeness and effectiveness in the development of a system.

REFERENCES

Chestnut, Harold, "*System Engineering Tools*". John Wiley & Sons, Inc., New York, 1965.

Comer, E.R., "*Catalyst: Automating System Engineering in the 21st Century*", Proceedings of the Second Annual International Symposium of the National Council On Systems Engineering (NCOSE), Seattle WA, 20-22 July 1992.

LaMonica, F., "*System Engineering Concept Demonstration Overview*". Proceedings of the Second Annual International Symposium of the National Council On Systems Engineering (NCOSE), Seattle WA, 20-22 July 1992.

Ortiz, A., "*SECD Process Model, Final Technical Report*, Volume 3, Rome Laboratory, Griffiss Air Force Base NY, May 1992

Ortiz, A., "*SECD Process Model Supporting Document*, Final Technical Report, Volume 3, Rome Laboratory, Griffiss Air Force Base NY, May 1992

AUTHOP'S BIOGRAPHY

Mr. Alberto Ortiz is a Senior Engineering Scientist at McDonnell Douglas Corporation - Douglas Aircraft Company (MDC-DAC). He is a technical staff to the Flight Controls Systems Group Leader. He is the Principal Investigator in the System Engineering Concept Demonstration (SECD) program and supports various advanced design concepts studies and proposals. He worked in the B-2 Avionics Integration Laboratory (AIL) in the development of requirements for the Defense Management System (DMS). He worked in the integration, testing and formal qualification of the UHF Transition MILSTAR Terminal.

Mr. Ortiz received a B.S. degree in Engineering from Southern Illinois University in 1984. He served three years in the U.S. Army and four in the U.S. Air Force. He is an IEEE and NCOSE member.

SYSTEM ENGINEERING OF NAVAL AIR ASW SYSTEMS

Richard J. Pariseau
Antisubmarine Warfare Systems Department

Henry G. Stuebing
Systems and Software Technology Depar.ment

Naval Air Warfare Center Aircraft Division Warminster
Warminster, PA 18974-5000

Abstract. This paper describes system engineering of Air Antisubmarine Warfare (ASW) Systems at the Naval Air Warfare Center Aircraft Division Warminster. These Combat Systems are examples of time-critical airborne applications that use distributed computer configurations. Structured interviews with the responsible system engineers were conducted to map working level processes(e.g. the time-line method) into the formal Navy processes. The paper reports the results of these interviews. The paper concludes with an identification of areas where automation has the potential of significantly improving the system engineering process.

BACKGROUND

The Naval Air Warfare Center Aircraft Division Warminster (NAWCADWAR), formerly the Naval Air Development Center (NADC) is the Navy's principal research and development laboratory for aircraft systems, air Antisubmarine Warfare (ASW) and navigation. This site is involved in all the key technology areas associated with ASW, Tactical Aircraft, Command Control and Communications, and Battleforce Integration. For Air ASW, we perform Warfare Analysis and System Concept Formulation, define Avionics System Architecture and Weapon System Architecture, and develop Engineering and ASW Support Systems. We provide a full spectrum of scientific and engineering skills across the system life cycle. Primary emphasis is in the areas of concept exploration and definition, demonstration and validation, and engineering and manufacturing development. NAWCADWAR has contributed to thirty years of evolution of air ASW in the areas of Land Based Patrol, Carrier Based Fleet Defense, Carrier Inner Zone Defense, and ASW Pickett Convoy Defense.

NAWCADWAR currently is organized into seven technical departments responsible for air ASW, tactical air, and the supporting scientific and engineering technologies. The Antisubmarine Warfare Systems Department (ASWSD) is responsible for ASW programs/projects, tasks and related support systems. These include systems engineering facilities, software development, generation and life cycle support facilities, and platform/sensor simulation and system performance prediction laboratories. The ASWSD is especially focused on program/project management, system acquisition and system engineering.

The Systems and Software Technology Department (SSTD) is responsible for the software engineering aspects of systems acquisition, software systems research, advancing software and systems engineering technology, and the development of tools and simulations. The SSTD is especially focused on software engineering solutions to systems acquisitions.

NAWCADWAR was invited to collaborate with Rome Laboratory in order to add a Navy ASW system engineering perspective to the Air Force Command, Control and Communication system engineering perspective. A joint U.S. Air Force - U.S. Navy Memorandum of Agreement was established in June 1990 in order to provide the vehicle for that collaboration. Please see (Lamonica 92).

INTRODUCTION

Naval Systems. The Department of Defense (DoD) is concerned with national security and conducts many acquisition programs in support of those concerns. All systems acquired by DoD are defense systems. In practice the acquisition programs are managed by the Services to support their respective missions. Clearly, the term defense system covers a wide spectrum of systems.

The Department of the Navy (DON) has certain mission responsibilities to support the goals of DoD. To fulfill those responsibilities the DON acquires many Naval systems. Like "defense system," the term "Naval system" covers a wide spectrum of systems. A Naval Air ASW System falls in the general class of combat systems; it is further characterized by having multiple sensors and weapons, several crew members, and long complex missions. The associated computer systems are time–critical applications on distributed computer configurations.

System Engineering Definition. We use the following definition (DSMC–90):

Systems engineering is the management function that controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process that transforms an operational need into a description of system parameters and integrates those parameters to optimize the overall system effectiveness.

The key systems engineering tasks are identified as (DoD–91):

1. Translating operational requirements into design requirements.

2. Transitioning technology from the technology base to program specific efforts.

3. Establishing a technical risk management program.

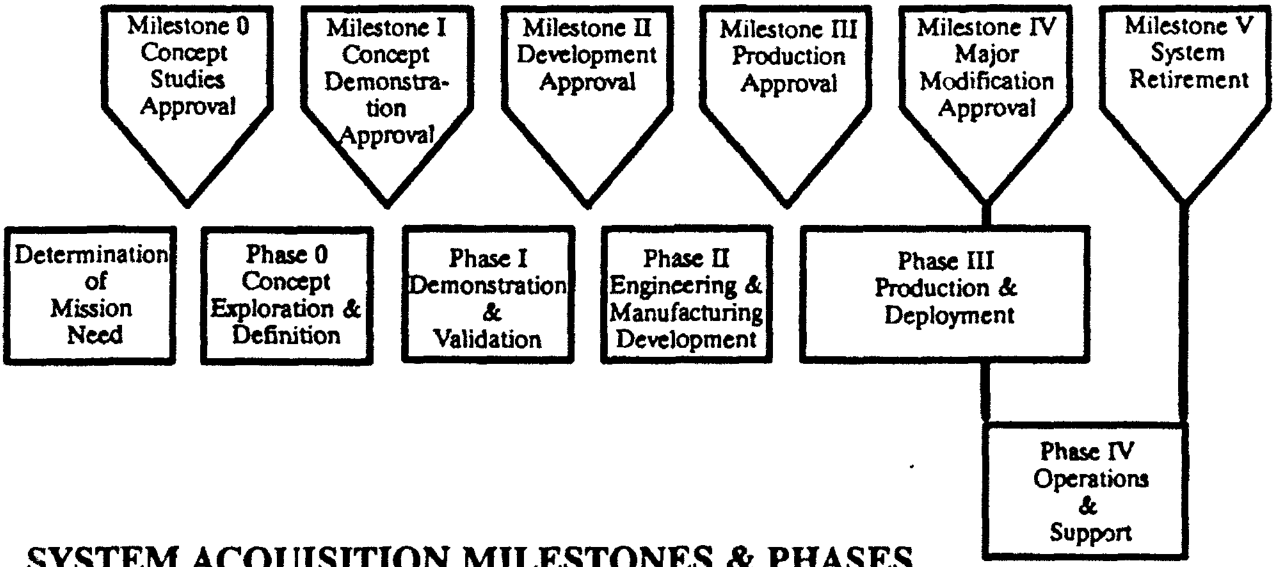
4. Verifying that the system design meets operational need.

SYSTEM ENGINEERING IN THE CONTEXT OF THE NAVY AIR ASW SYSTEM LIFE CYCLE

Introduction. A full discussion of system engineering of defense systems is beyond the scope of this paper. The Department of Defense (DoD) and the Department of Navy (DON) provide numerous descriptions of systems engineering. In particular, (DoD-91), (DSMC–90), (DON–89), and (AIR–87) provide detailed descriptions of all the possible system engineering tasks associated with major defense system acquisitions. In practice, at field organizations such as NAWCADWAR, the actual systems engineering tasks are a subset of the full set described by the references. The purpose of this section is to describe the typical system engineering tasks of a Navy field organization within the context of the DoD system life cycle.

Acquisition Milestones and Phases. The five major milestone decision points and the six phases of the acquisition process are shown in Figure 1. These milestones and phases are thoroughly discussed in the references and are summarized below:

Determination of Mission Need. DoD acquisition programs result from identified mission needs. These needs result from assessments of current and future threats, current and projected capabilities, and the context of national defense policy. The mission need may establish a new operational capability or may improve an existing capability. Various documents result from this phase. The Tentative Operational Requirement (TOR) describes the desired capabilities of the proposed system in general terms. The Development Options Paper (DOP) explores optional approaches to meeting the desired capabilities of the proposed new system. Once the system approach is selected, it is documented in an Operational Requirement (OR) or, for special acquisition categories, in a Mission Need Statement (MNS). The issuance of the OR or MNS commits the Chief of Naval Operations Resource Sponsor to support the new program both in the Program Objectives Memorandum (POM) and the budget process. Secretary of Defense approval of the program occurs with the Acquisition Decision Memorandum (ADM). This is the Milestone 0 decision shown in Figure 1. If the budget process results in program funding, the next phase is initiated.


Milestone 0 Concept Studies Approval
Milestone I Concept Demonstration Approval
Milestone II Development Approval
Milestone III Production Approval
Milestone IV Major Modification Approval
Milestone V System Retirement
Determination of Mission Need
Phase 0 Concept Exploration & Definition
Phase I Demonstration & Validation
Phase II Engineering & Manufacturing Development
Phase III Production & Deployment
Phase IV Operations & Support

SYSTEM ACQUISITION MILESTONES & PHASES
FIGURE 1

Concept Exploration and Definition (Phase 0). This phase consists of competitive, parallel short term studies by the DoD and/or industry. It evaluates alternate concepts from the viewpoint of life cycle cost estimates relative to the value of the increased operational capability. The requirements for the system are analyzed and a system requirements specification is generated. Potential architectures and designs are considered with engineering analyses and trade studies. Results include life cycle cost estimates, trade-off studies, critical system characteristics, operational constraints, and support requirements. The conceptual design of the defense system is complete, but only at the highest levels. This is used by the DoD to create a draft version of the System/Segment Specification (SSS) and the Test and Evaluation Master Plan (TEMP). Depending upon the acquisition category a System Concept Paper (SCP) or a Decision Coordinating Paper (DCP) may be produced to formally document the phase results for the Milestone I decision. The ADM for Milestone I confirms that there are valid threat assessments and performance objectives, that the studies support the need, that the program is affordable over its life cycle, and that adequate resources exist or can be obtained. It authorizes the next phase.

Demonstration and Validation (Phase I). The purpose of this phase is to validate the system requirements, to better define critical design characteristics, to demonstrate that critical technologies can be used, to prove that critical processes are attainable, and to demonstrate that the architecture and design are sufficiently understood to continue to engineering and manufacturing development. The results of this phase are a validated SSS which becomes the system Functional Baseline, and the identification of all configuration items as part of the Allocated Baseline. The TEMP is updated and a Systems Engineering Management Plan (SEMP) is produced. A DCP is prepared. The ADM for Milestone II confirms that optimum technical and support approaches have been selected, that technology requirements are available, that risks are acceptable, and that the cost is affordable. It authorizes the next phase.

Engineering and Manufacturing Development (Phase II). The purpose of this phase is to translate the selected Phase I system specification into a stable, producible, and cost effective system design. That design will be implemented and validated through testing. The manufacturing or production process will also be validated. Due to the size of most DoD systems, this phase is usually contracted to industry. The operational requirements are translated into the desired system specification, contractual actions are initiated, and the best proposal(s) are selected. A version of the system is constructed to prove that all system requirements will be met. The proof is obtained by formal and independent Operational Testing and Evaluation (OPEVAL). The results of this phase are a Product Baseline which contains the detailed design documentation and manufacturing requirements necessary to go to production and the Integrated Logistic Support documentation necessary to field and fully support the system. A DCP is prepared. The ADM for Milestone III authorizes production and deployment.

Production and Deployment (Phase III). This phase overlaps with the Operations and Support phase as shown in Figure 1. The phase consists of two concurrent activities. Production begins with production approval and ends when the last system is delivered; deployment begins with the delivery of the first system to the field and ends when the system is retired from the operational inventory. The objectives of the phase are to establish a stable production and support base and to verify production and operational performance through follow-on testing. During this phase, changes in threats or Defense Planning Guidance, deficiencies identified during follow-on testing or actual use, or cost reduction opportunities may lead to proposed major modifications. These major changes to the system are authorized by the ADM for the Milestone IV decision. This decision determines if the major modification is warranted and establishes an approved acquisition strategy. This strategy may require the acquisition to begin again at any of the acquisition phases (0, I, II, or III).

Operations and Support Phase (Phase IV). This phase overlaps with the Production and Deployment phase. *It occurs after initial* systems have been delivered to the field. The phase usually begins when management responsibility for the system is transferred from the developer to the maintainer and ends when the system leaves the inventory. Quality and safety problems are corrected and shortcomings and deficiencies are identified for possible performance improvement.

Field Activity Systems Engineering. Within the DON, Naval Air Systems Command (NAVAIRSYSCOM) air ASW acquisitions are usually managed by Program Executive Officers and Program Managers who obtain technical support from NAVAIRSYSCOM technical divisions. Due to resource limitations, those divisions task various field activities to provide specific expertise and additional personnel in order to form the Navy technical team required for successful acquisition. In particular, tasking for Naval Air ASW Systems is directed to the NAWCADWAR.

NAWCADWAR Systems Engineering. Within the NAWCADWAR, the ASWSD is responsible for efforts related to air ASW systems. The ASWSD is organized into four product lines: Land Based Fixed Wing Maritime Patrol ASW Aircraft, Carrier Based Fixed Wing ASW Aircraft, Vertical Flight ASW Aircraft, and Advanced ASW Systems and Sensor

Integration. The ASWSD produces products for the Fleet when such efforts fall within its resource constraints. More typically in Air ASW Systems, the ASWSD serves as the technical arm of the NAVAIRSYSCOM offices responsible for major system acquisitions. At NAWCADWAR, technical specialists are located across 4 technology departments that support the 3 warfare area departments. In solving the system engineering and product development problem, the system engineers and project engineers form teams of experts from the technology departments.

Systems Engineering Tasks. The systems engineering tasks for all four air ASW product lines in the ASWSD are very similar. The following summation is based on systems engineering tasks in the Vertical Flight Program Division (VFPD) of the ASWSD.

In the VFPD, systems engineers may help in creating the DOP but usually become involved in the acquisition process after the Milestone 0 decision. During Phase 0, the systems engineers use an approved OR or MNS as the basis for the preliminary system specification. They may support the NAVAIRSYSCOM in the development of the SCP or DCP and the TEMP. For Phase I, systems engineers provide support in identifying the preferred system concepts to be tested, identifying risk areas, and evaluating experimental results. During this phase, they begin work on the final system specification, the interface specifications, and test requirements that will be needed early in the next phase and they might support NAVAIRSYSCOM in updating the TEMP and developing the SEMP and the DCP. The systems engineers provide support for NAVAIRSYSCOM during Phase II. This includes final translation of the operational requirements into the system specification, specification of system interfaces, and development of test requirements, plans, and procedures. They perform contractor monitoring during system development to insure that there is an optimum balance of all system elements and that the integration of the system optimizes the overall system effectiveness. During testing, the systems engineers monitor the contractor's development testing and participate in the Navy's independent technical evaluation. Finally, the system engineers provide NAVAIRSYSCOM with the support required to respond to criticism resulting from the Navy's independent OPEVAL of the system. In Phase III, the systems engineers provide support in addressing operational problems and major modifications and support development of the Engineering Change Proposals required to implement major modifications. By Phase IV, system change is usually minor and the systems engineers no longer have significant involvement.

FIELD INTERVIEWS AT NAWCADWAR

Background. To determine common systems engineering problems and issues, a survey of studies and reports dealing with the life cycle of mission critical systems was performed. To augment the results of this study, field interviews were performed by Mr. E. Comer and Mr. A. Ortiz in August 1990 with 3 system engineers and 4 project engineers belonging to the Vertical Flight Program Division (VFPD) of the ASWSD at NAWCADWAR. These interviews are described in (Rome-91).

Systems Engineers. Systems engineers in the VFPD are senior engineering personnel with significant knowledge and experience in Air ASW systems. They are the lead technical person for product developments and represent the NAWCADWAR on various Navy committees and working groups. They have a graduate degree in electronic engineering or physics or equivalent experience. Their career develops from that of an engineer working on specific components and subsystems through increasing technical leadership and finally to the position of systems engineer. The interviewed systems engineers had a total of 70 years of experience. Two had been key participants in the development of the Navy's Helicopter ASW capability. They have wide ranging functional knowledge in such areas as acoustics, simulation, analysis, and the design of avionic systems. They also have detailed knowledge of the Navy's system acquisition process.

Project Engineers. Project engineers in the VFPD are senior engineering personnel responsible for technical management of the NAWCADWAR team developing or supporting the Navy development of an Air ASW System. They have between 5 and 15 years of experience as engineers and lead engineers working on components and subsystems. They have a graduate degree or are in the process of acquiring one. The interviewed project engineers had a total of 35 years of experience and had been project engineers for about 2 years. Their technical experience includes avionics software, electronics, system integration, and testing. One had performed as a systems engineer. They have detailed knowledge of the Navy's system acquisition process.

Systems Engineering Tasks. In the VFPD, systems engineers usually are involved in the acquisition process during Concept Exploration and Definition and continue to provide support for NAVAIRSYSCOM through Production and Deployment. Their typical tasking has been described earlier in this paper.

Project Engineering Tasks. In the VFPD, projects are usually formed at Demonstration and Validation (Phase I) and continue through

Production and Deployment (Phase III). During these phases, the project engineers manage the technical teams that support NAVAIRSYSCOM in the systems acquisition.

Automation Support. In the VFPD, systems engineers and project engineers have the following automation support: Personal computers on each desk and avionic integration and testing facilities for specific avionic systems and subsystems. A Systems Engineering Facility (SYEF) is being developed.

Personal computers are usually IBM clones built around a 286 or 386 processor. They contain software for word processing, spreadsheets, databases, graphics, project management, and E-mail. Electronic networking exists within NAWCADWAR and to some other sites (primarily NAVAIRSYSCOM). A Program Database System with electronic linkage to both internal and external organizations is currently being developed by the VFPD and is in the initial testing phase.

The avionic integration and test facilities vary from simple (interface stimulation, control, and testing) to complex (full avionic system with both stimulation and simulation capability). The facilities can be used for analysis and concept exploration. Also available are engineering laboratories (e.g., acoustics, displays, mechanical, composites, communication, and radar) which support engineering analysis for systems engineering decisions.

The SYEF currently consists of SUN4s (260, 280, 330, and 75GX) and associated peripherals. Software includes publishing software, relational database software, requirements tracing software, design software, and software being evaluated to support systems engineering (Lefkovitz-91). The SYEF is in the process of developing an integrated system engineering environment. The SYEF will be a test site for products developed under the SECD.

Interviews. The current product area of the interviewed personnel is helicopter ASW avionics systems. However, most of the interviewed personnel had both knowledge and experience in the other product areas.

Objectives. The interviews had three objectives: (1) Understand the practiced systems engineering process as opposed to the theoretical systems engineering process; (2) understand the variability in the process across systems engineers; and (3) understand the systems engineering automation areas that would be considered high priority by systems engineers.

Approach. The approach chosen was a type II interview as defined in (Bouchard-79). This is a structured interview having specified questions but leaving the character of the response open. The questions were broad and allowed the interviewees to express themselves freely. There were seven specific questions dealing with important issues that were spaced throughout the interview. These were:

1. What are the most difficult aspects of building these kinds of systems?
2. What do you consider the most important aspects of your job - the things that are most critical to having a successful system?
3. What activities consume most of your time? What is the "pie chart" describing what you do?
4. What are the most difficult aspects of your job? How are they difficult?
5. If you were training someone in your job, what would you teach? What advice would you give?
6. If you could improve any aspect of your job, what would it be?
7. What automation tools or aids do you wish you had?

These questions were inserted in each 90 minute interview that covered 12 areas dealing with the interviewee profile, the application area, the process, roles and responsibilities, individual activities, methods, and automation, interaction, organization connectivity, classified information, and improvement in method and automation. The 2 interviewers asked the questions and took notes. The interviews were audio taped for later reference. Following the interviews, the data was reduced and analyzed. For a complete description of the process see (Rome-91).

Results. The following was observed:

Growth in Complexity. The interviews indicated that there is a trend for the Air ASW Systems to get increasingly complex. The number of processors in the avionic systems is increasing as is the size of the software. Desires to minimize crew size are resulting in increased dependence on improved displays and controls and software that supports crew decisions for all aspects of the mission. The wide range of technology and function appearing in the avionic systems for Air ASW is forcing systems engineering to place greater reliance on subsystems engineering and technical specialists. Imposed schedules require concurrent approaches to the subsystem engineering and the systems engineering. The net result is a need for processes and tools that support rapid and accurate coordination of complex concurrent engineering efforts.

Personnel Characteristics. Systems engineers and project engineers had a great amount of application experience. They had spent most of their career in the same application. They had started at the bottom and worked their way up through hands on experience. They were now mentors in the organizations. In general they were eclectics and generalists who were open minded, assertive, and provided leadership.

Priority Areas. Their priority areas were requirements engineering, collaboration anu oordination, scheduling, interface management, nd the applicability of standards and policies.

Requirements engineering included the effort f converting operational requirements into system equirements, decomposition and allocation of unctional requirements into the allocated baseline, racking the requirements through the system life ycle, and handling change.

Collaboration and coordination dealt with the roblem of bringing together the multi-functional eam for development of (or major modification to) he system requirements.

Scheduling required dealing with constraints, naintaining proper priorities, interfacing with arious reporting systems (internal and external to AWCADWAR), and effective time management.

Interface management had problems of lefinition and change similar to requirements ngineering.

Applicability of standards and policies had to do vith identifying the standards and Data Item escriptions required for contracts.

Personal Activities. While the answers on their ersonal activities varied, they appeared to be ffected by differences in viewpoint and terminology. The systems engineers thought they spent significant ime in coordination, engineering, analysis, and locumentation. Project engineers thought they pent significant time in coordination, locumentation, and planning.

Methods. There was no clear definition of the nethods applied to systems engineering. nterviewees stated that large amounts of nformation were used, reviewed, and generated. nformation was located by knowing where to look or y asking a more experienced individual. Typical inds of information used and produced were block liagrams, time-line diagrams, hierarchy of functions, unctional flow diagrams, system decomposition, lternatives analysis, trade studies, specific analysis, nodels, and critical parameter tracking.

Interaction. There was a high level of nteraction with both internal and external ersonnel. The systems engineer and the project ngineer worked closely together. Both had frequent nteractions with the customer. Both had frequent nteractions with internal supporting organizations. oth frequently performed as mentors. Since both he scope of the efforts and the schedule required eam efforts, work involved concurrent approaches vith multi-functional teams. The team oriented nteractions resulted in little conflict. Interaction was nost frequently by voice and paper. External nteractions frequently used facsimile. E-mail ommunication was increasing but could not be used or formal communication due to the policy for uthorized release of information.

Use of Automation. The use of automation by systems engineers and one project engineer was limited. The other project engineers were frequent users. The difficulties in the former case were:

A basic systems engineering task is the transformation of operational requirements to system requirements. Operational requirements regardless of whether they are being described in a TOR, an OR, a DOP, a SCP, or a DCP are frequently classified. Much of the information produced by the systems engineers is also classified. This could include system performance parameters, system components, test requirements, plans, and procedures, algorithms, and even information concerning recipient platforms. The DoD regulations governing the handling of classified information on computers and in electronic transmission severely restrict the systems engineers in their use of automation.

The systems engineering job is highly interactive. Face to face meetings and secure telephone communication are quicker and easier than attempting to use secure electronic media (facsimile or E-mail).

The interviewed systems engineers and one project engineer were not very computer literate. This appears to be more a result of infrequent use due to the classified information problem than reluctance or inability to learn. Training, both initial and refresher, has been provided. However, the systems engineers have much less opportunity to use the automation available then do other engineers.

Conclusions. The summary conclusions from the interviews are:

1. Both the systems engineers and the project engineers had heavy workloads in tasks that could benefit from automation.

2. The automation should address the following:

The use and manipulation of requirements throughout the system life cycle.

The problem of collaboration/coordination of multi-disciplined teams across both the internal and the external organizations.

Support for scheduling issues.

The definition, management, and control of system interfaces.

Support for locating, determining, and applying standards and policies.

3. However, unless the automation addresses the problem of classified information, the use of the automation will be severely limited.

TIME-LINE SYSTEMS ENGINEERING METHOD

We use the following as our working definition of our process: The process by which operations and engineering are performed to ensure that a given design meets detailed mission functional and performance requirements within the program cost constraints, including the procedures to track the mission effectiveness of the system throughout development.

We have evolved a heuristic process over the last 15 years and have successfully applied it to several systems; the process is purely manual. The process has 6 steps that are briefly described in the following paragraphs; locally, it is called the "Time-Line Method".

1. Detailed Mission Scenario Preparation. In the first step a team of variable size, led by a systems engineer, constructs paper scenarios of battles or missions for a proposed or modified system. These scenarios cover mission preparation, conduct of the mission, and post mission activities. Preparing the scenarios is an information-gathering intensive activity for the team. The products of this step are scenario diagrams and associated descriptions.

2. Mission Functional/Performance Analysis And Definition. The second step is generally performed by a small group of systems engineers and consists of identifying the functions for each relevant time and event shown on the mission scenarios. This allows the establishment of system performance requirements. This also provides a traceable path from a mission requirement, as shown on the scenario, to a system function. The product of this step is a set of functional analysis diagrams and associated descriptions.

3. Subsystem Definition And Allocation. The third step, which is sometimes called Subsystem Conceptual Design, is performed by a team of subsystem specialists that is led by a systems engineer; the team size runs from 8 to 12 people. In this step all subsystems required by the scenarios and the analysis of step 2 are identified. Subsystem specialists determine what specific equipment has sufficient design maturity to satisfy the requirements. The selected subsystems are grouped into an overall conceptual design with functional interfaces. The primary activity in this step is the allocation of functions to subsystems. These activities imply the evaluation of alternative designs and the identification of design risk. This step typically takes 4–6 months and produces conceptual schematic block diagrams.

4. Hardware/Software/Operator Function Partitioning. In the fourth step the scenarios, functional analyses, and conceptual designs are used to determine the detailed subsystem functions, including the inputs and outputs of equipments and functions. Required duration of functions and time-dependencies between subsystems are identified. This allows an early estimation of system throughput. Also, human factor specialists assess the work load on the system operators. The products of this step are operational sequence diagrams and technical data that form the basis for the system functional specifications. The operational sequence diagrams or time-lines are the source of the informal name of our system engineering process.

5. Functional Design. In the fifth step the system architecture and design are accomplished. Specialists from all applicable technologies participate under the leadership of a systems engineer. The functional requirements are evaluated and system effectiveness trades are conducted. The hardware is selected. Core subsystems that are common to multiple missions are identified as well as unique subsystems that are required for specific missions.

6. Specification. The sixth and final step consists of producing specifications in accordance with the applicable military specification standards. These specifications are typically functional in nature and include detailed functional interfaces. Core subsystems are addressed as well as unique mission specific subsystems. An emphasis is placed on integrated subsystem designs.

NEED FOR AUTOMATION

Problem. The systems engineering problem at NAWCADWAR is how to increase productivity and improve quality when weapon systems are becoming increasingly complex and defense budget constraints limit personnel increases. Currently, the systems engineering methods are manual and informal. They require large amounts of manpower and offer scope for error. The increasing complexity of the weapon systems requires the collaboration and coordination of teams of technology specialists, and this process increases the workload on the systems engineers. Since skilled systems engineers require years of professional growth, the problem cannot quickly be solved by internal personnel development. A possible solution is to provide our current systems engineers with automation that will decrease their workload, increase their productivity, and support them in preventing error.

Technical Issue. A weapon system consists of a network of hardware, software and human operators. The hardware and software elements are typically associated with a several interactive subsystems. The man-machine interface includes switches and keyboards for input, and visual, audible, or hard-copy output. The systems engineer's role is to combine these parts into a cohesive system that fulfills a set of prescribed requirements. Our current systems engineering practice consists primarily of informal manual methods and procedures. Formal systems engineering methods exist that reduce the possibility that certain steps or activities are overlooked, and also force a rigorous, organized and structured way of approaching, segmenting, and solving a system design problem. Manual execution of a formal method, however, can be a tedious and difficult job.

Computer based tools that support a formal method can off-load the systems engineer from repetitive and tedious tasks, and also provide cross-checking and acing functions that would overload the human ind. The technology with which current and future ystems will be built is itself changing at a rapid rate; omputer based tools also offer the ability to quickly hange the system model in response to the rapidly hanging implementation-technology, allowing the ystems engineer to make better decisions quicker.

nother benefit of formal methods with computer ased tools is that the system testing can be done ore thoroughly. This is because the system pecification is developed more precisely and ccurately, thus allowing the test requirements to be ore completely specified.

omputer based tools available to the systems ngineer range from sophisticated tool sets and pecification languages that guide him through the ormal methods, to simpler tools that provide word rocessing and documentation support, graphics and lotting, and communication (electronic mail).

everal barriers exist that prevent or slow the olution process. First, is the lack of clear examples hat show the benefit of the new systems engineering echnology. Second, is a precise estimate of the apital costs that would be required to use the new echnology. Third, is obtaining the management ommitment that would be required to train the urrent work force in the new methods and manage he transition to actual use. Fourth, is the problem of andling classified information.

major consideration for the future is coordination etween the Services. Rather than an environment vhere the major emphasis was on the rapid levelopment and deployment of many systems, we re now in an era where efficiency and economy are rimary factors. Systems engineering spans both echnical and economic dimensions. Automation eems to offer an attractive opportunity both for ystems engineering and all of the associated peciality engineering. A significant effort has been tarted by the U.S. Air Force's Rome Laboratory with heir System Engineering Concept Demonstration SECD); we believe this was an important and ignificant step and look forward to future oordination and cooperation.

REFERENCES

ouchard, Thomas J., "Field Research Methods: nterviewing, Questionnaires, Participant bservation, Systematic Observation, Unobtrusive Measures," *Handbook of Industrial and Organizational sychology*, Marvin D. Dunnette (ed), Rand McNally ollege Publishing Company, Chicago, IL, 1976. 63–413

epartment of Defense Directive Number 5000.1 of 3 February 1991

Department of the Navy, "RDT&E/Acquisition Management Guide, 11th Edition, January 1989

Defense Systems Managᵉment College, "Systems Engineering Management Guide", January 1990

Lamonica, F.S., "System Engineering Concept Demonstration - An Overview", *Proceedings of the Second Annual International Symposium of the National Council on Systems Engineering (NCOSE)*, Seattle WA 20-22 July 1992

Lefkovitz, David, "Investigation into the Use of STATEMATE and TAGS as Real-Time System Specification Languages", Final Report for Vertical Flight Program, Naval Air Development Center, 30 October 1991

Naval Air Systems Command, Systems Engineering Group Instruction 5451.2 of 21 September 1987

Rome Laboratory, Griffiss Air Force Base NY, "System Engineering Concept Demonstration (SECD) Final Technical Report", draft, Contract Number F30602-90-C-0021, May 1991

AUTHORS BIOGRAPHIES

Richard J. Pariseau is the Vertical Flight Program Engineer at the NAWCADWAR. He is currently responsible for the management of NAWCADWAR efforts on 7 Helicopter ASW Projects ranging from new development efforts to life cycle support of deployed systems. He has 32 years of experience in Naval computer systems, software engineering, simulation, software research, and the development of air and ship based defense systems. He received a B.S. degree in Physics from the Drexel Institute of Technology in 1964. He is a member of the IEEE Computer Society, the ACM, and the AIAA. He is currently serving on the AIAA Technical Committee for Software Systems.

Henry G. Stuebing received the B.S. degree in physics and mathematics from Ursinus College in 1958; after completing graduate work in physics he studied computer engineering in the Moore School of Electrical Engineering of the University of Pennsylvania and received the M.S. degree.

From 1983 to 1986 he was Chairman, STARS-SEE (Software Engineering Environment) Tri-service Committee.

In 1985 he received the Department of Navy Superior Service Medal for his work in Software Engineering Environments.

His current position is Senior Technical Consultant to the Director of the Systems and Software Technology Department where he advises Department and Center projects on system and software issues.

Mr. Stuebing is an Associate Fellow of AIAA, and a member of ACM and IEEE: he also serves on various Navy committees.

References

[AFS89] Air Force Studies Board. *Adapting Software Development Policies to Modern Technology*. Washington: National Academy Press, 1989.

[BEA90] Beam, Walter R.. "Systems Engineering Activites Design of C-Cubed Systems." Unpublished paper. September 11, 1990.

[BOU76] Bouchard, Thomas, J., "Field Research Methods: Interviewing, Questionaires, Participant Observation, Systematic Observation, Unobtrusive Measures," *Handbook of Industrial and Organizational Psychology*, Marvin D. Dunnette (ed), Rand McNally College Publishing Company, Chicago, IL, 1976. 363-413.

[BUC90] Buckley, Fletcher J. "Standards." *Computer*. September 1990. 82-84.

[CHI88] Chin, Janet S. "New Procedures for Graphics Standardization." *IEEE Computer Graphics and Applications*. Volume 8. Number 6. November 1988.

[COD84] Council of Defense and Space Industry Associations (CODSIA). *DoD Management of Mission-Critical Computer Resources*. CODSIA Report 13–82 (Volume I, Task Group Report) to Under Secretary of Defense, Research and Engineering. March 1984.

[COH83] Cohen, Danny, and Jon Postel. "The ISO Reference Model and Other Protocol Architectures." Information Sciences Institute, ISI Reprint Series, ISI/RS-83-6, November 1983.

[COM87a] Comer, Edward R., and J. Kaye Grau. "SEE Information Interface Analysis." Software Productivity Solutions, Inc. January 1987.

[COM87b] Comer, Edward R., and J. Kaye Grau. "Database Organization for Software Engineering Environments." Software Productivity Solutions, Inc. January 1987.

[CUR87a] Cureton, Bill. "The Future of UNIX As A Platform for CASE." *First International Workshop of Computer-Aided Software Engineering*. Volume 1. Cambridge, Massachusetts. May 27-29, 1987. 211-215.

[CUR87b] Curtis, Bill. "Introduction to Empirical Research on the Design Process in MCC's Software Technology Program." *Empirical Studies of the Design Process: Papers for the Second Workshop on Empirical Studies of Programmers*, MCC Technical Report Number STP-260-87, Sept. 24, 1987. 1-4.

[FOR89a] Forte, Gene. "Rally Round the Repository." CASE Outlook. Volume 89, Number 2. 1989. 5-10.

[FOR89b] Forte, Gene. "DEC's CASE Counterpunch." CASE Outlook. Volume 89, Number 2. 1989.

[HAY89] Hayes, Frank. Baran, Nick. "A Guide to GUI's." *Byte*, July, 1989. 250-257.

[HOU87] Houghton, Raymond, Jr. and Dolores Wallace. "Characteristics and Functions of Software Engineering Environments: An Overview." *ACM Software Engineering Notes*. Volume 12. Number 1. January 1987.

[JON90] Jones, Randy. "Nets Are The Name Of The Game." *Unix Today*. September 17, 1990.

[KRA87] Krasner, Herb, et. al. "Communication Breakdowns and Boundary Spanning Activities on Large Programming Projects." *Empirical Studies of the Design Process: Papers for the Second Workshop on Empirical Studies of Programmers*. MCC Technical Report Number STP-260-87, Sept ember 24, 1987. 46-66.

Also published in *Empirical Studies of Programmers: Second Workshop*, G. Olson, S. Sheppard, and E.Soloway, eds., Norwood, N.J.: Ablex Publishing Corporation, 1987.

[KRA88] Krasner, Herb, et. al. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, Vol. 31, No. 11, November 1988.

[KOC86] Kochan, Stephen G., and Patrick H. Wood. *Exploring the UNIX System*. AT&T Bell Laboratories, Whipany, N.J. 5-6.

[LIV90] Livingston, Dennis. "Here come object-oriented databases!" *System Integration*. July 1990 5058.

[MAG90] Maguire, John. "The Switch to Open Systems." *Unix Today*. September 17, 1990.

[NAS85] Nash, Sarah H., and Samuel T. Redwine, Jr. "Information Interface Related Standards, Guidelines, and Recommended Practices." JSSEE (Joint Service Software Engineering Environment), SEE-INFO-004, IDA Paper P-1842, July 1985.

[NAS86] Nash, Sarah H., and Samuel T. Redwine, Jr. "A Map of the World of Software-Related Standards, Guidelines, and Recommended Practices." Proceedings Computer Standards Conference 1986, May 13-15, 1986. San Francisco, CA. 136-159.

[PET87] Peterson, James L. and Abraham Silberschatz. *Operating System Concepts*. Addison-Wesley Publishing Company. 1987. p. 39-56.

[REP89] *Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation*, Staff Study by the Subcommitte on Investigations and Oversight transmitted to the Committe on Science, Space and Technology, U.S. House of Representatives, September 1989.

[RUD89a] Rudmik, Andres. "Integration Information Systems." Wright-Patterson Air Force Base, Logistics and Human Factors Division. July 1989.

[RUD89b] Rudmik, Andres. "Environment Integration Technology." Naval Air Development Center (NADC), Contract No. N69-86-C-0415, Software Productivity Solutions, Inc. 1989.

[SIS86] Sisson, Norwood. "Dialog Management Reference Model," SIGCHI Bulletin. October, 1986. 34-35.

[SMI82] Smith, S.L. "User-system interface." Human Factors Society *Bulletin*. Volume 25. Number 3. 1.

[SMI86] Smith, S. L. and Jane N. Mosier. "Guidelines for Designing User Interface Software." MITRE Corporation. Electronic Systems Division, AFSC. Hanscom AFB, MA. 1986.

[SOW84] Sowa, J.F., Conceptual Structures, Information Processing in Mind and Machine. Addison-Wesley Publishing Company, Inc., 1984.

[ST90a] Staff. Honeywell. " Data Repository." *EIS Update.* Volume 2. Number 9. April, 1990.

[ST90b] Staff. *Newsgram*. McDonnell Douglas. Information Systems Engineering. June 1990.

[ST90c] Staff. Ada Information Clearinghouse Newsletter, *AdaIC.* Vol. VIII, No. 3. September, 1990.

[ST90d] Staff. Ada Information Clearinghouse Newsletter, *AdaIC.* Vol. VIII, No. 2. June, 1990.

[ST90e] Staff. The Postscript Industry Newsletter. *PostScript Language Journal*. Volume 2. Number 3, 1990. 2.

[ST90f] Staff. CADENCE. Product Announcement. Cadence Design Systems, Inc. SanJose, CA. 1990.

[STA85] Software Engineering Environment Standard Interfaces. SEE-INFO-005. September 16, 1985.

[TER90] Terry, B., and D. Logee. "Terminology for Software Engineering Environment (SEE) and Computer-Aided Software Engineering (CASE)." *ACM Software Engineering Notes*, Volume 15, Number 2, April 1990. 83–95.

[USA87] United States. Department of Commerce. Federal Information Processing Standards

[WEL89] Welke, Richard J. "Meta Systems on Meta Models." *CASE Outlook*. Volume 1. Number 4, 1989. 35-45.

Glossary of Symbols, Abbreviations, and Acronyms

In the following list of symbols, abbreviations, and acronyms, trailing parenthetical comments are used to identify responsible or otherwise associated companies/organizations/projects, and other informatory miscellany. Terms and symbols within square brackets are optional representations. Multiple definitions are separated by a semi-colon. Additional information on selected entries is provided in the Glossary.

AAE — Automated Access Experiment (SPS, RL)

AD/Cycle — Application Development/Cycle (IBM)

AD/SAA — Application Development/System Application Architecture (IBM)

AF — United States Air Force

AFS — Andrew File System

AFB — Air Force Base

AFR 800-14 — Life Cycle Management of Computer Resources in Systems (29 September 1986)

AFR — Air Force Regulation

AFSB — Air Force Studies Board

AI — Artificial Intelligence

AIAA — American Institute of Aeronautics and Astronautics

AJPO — Ada Joint Program Office (DoD)

ALICIA — Automated LIfe Cycle Impact Analysis System (SPS, RL)

AMS — Automated Measurement System (RL)

ANSI — American National Standards Institute

APDA — Apple Programmers and Developers Association

APSE — Ada Programming Support Environment

ARCS — Automated Reusable Component System (SPS)

ASCII — American Standard Code for Information Interchange

ASIA — Avionics/Electronics Subsystem Integration and Acquisition Project (MDC)

ASIC — Application-Specific Integrated Circuit

ASQS — Assistant for Specifying the Quality of Software (RL)

BS/OOD — Box Structures for Object-Oriented Development (SPS)

C^{3}I — Command, Control, Communication, and Intelligence

CAAPE — Computer Aided Avionics Project Environment (MDC)

CAD — Computer-Aided Design

CAE — Computer-Aided Engineering

CAIS — Common APSE Interface Set

CAIS-A — Common APSE Interface Set Revision A (militarized)

CALS — Computer-Aided Acquisition and Logistic Support (AF program)

CAM — Computer-Aided Manufacturing

CANDLE — Common Attributed Notation for IDL (University of North Carolina)

CASE — Computer-Aided Software Engineering

CBD — Commerce Business Daily

CDD — Common Data Dictionary (DEC)

CDIF — CASE Data Interchange Format

CDR — Critical Design Review (DoD–STD–2167A)

CDRL — Contract Data Requirements List (DoD–STD–2167A)

CEC — Commission of the European Community

CECOM — Communications-Electronics Command (U.S. Army)

CEF — Common Exchange Format (EIS)

CERC — Concurrent Engineering Research Center (DICE)

CFI — CAD Framework Initiative, Inc.

CFSR — Contract Funds Status Report

CIS — CASE Integrated Services

CLIN — Contract Line Item Number

CLOS — Common Lisp Object System

CM — Configuration Management

CMU — Carnegie Mellon University

CODSIA — Council Of Defense and Space Industry Associations

COEE — designation for Software Engineering Branch at RL (not an acronym)

COTS — Commercial-Off-The-Shelf

CR — Computer Resource

CRLCMP — Computer Resources Lifecycle Management Plan (AFR 800–14)

CRWG — Computer Resources Working Group (AFR 800–14)

CSC — Computer Software Component (DoD–STD–2167A)

CSCI — Computer Software Configuration Item (DoD–STD–2167A)

CSP — Cross Systems Product (IBM)

CSU — Computer Software Unit (DoD–STD–2167A)

DAC — Douglas Aircraft Company; Design Automation Conference

DACS — Data & Analysis Center for Software (RL, IITRI)

DARPA — Defense Advanced Research Projects Agency

DB — Database

DBMS — Database Management System

DCL — Digital Command Language (SLCSE)

DDBMS — Distributed Database Management System

DEC — Digital Equipment Corporation

DECNET — [re: communication protocol] (DEC)

DGL — Document Generation Language (SLCSE)

DIANA — Descriptive Intermediate Attributed Notation for Ada

DICE — DARPA Initiative in Concurrent Engineering

DID — Data Item Description (DoD–STD–2167A)

DIS — Defense Investigative Service

DOCGEN — Document Generator (SLCSE)

DoD — Department of Defense (U. S. Government)

DoD–STD–2167A — Defense System Software Development (revised 29 February 1988)

DOS — Disk Operating System; Distributed Operating System

DSEE — Domain Software Engineering Environment (Apollo)

DSI — Delivered Source Instructions

DSS — Distributed Simulation System (RL)

E[-]R — Entity[-]Relationship

ECAD — Electrical CAD (EIS)

ECP — Engineering Change Proposal (DoD–STD–2167A)

ECSAM — Embedded Computer Systems Analysis Method (MDC)

EDA — Electronic Design Automation

EDIF — Electrical Design Interchange Format

EES — Engineering Environment Services (EIS)

EIA — Electronic Industries Association

EIS — Engineering Information System (Honeywell Information Systems, AF)

EPAP — Electronic Product Automation Program (MDC)

ERIF — Entity Relationship Interface (SLCSE)

ESD — Electronic Systems Division (USAF)

ESSDA — Expert System for Software Design Analysis (SPS)

FAA — Federal Aviation Administration

FAR — Federal Acquisition Regulations (AFR 800–14)

FIPS — Federal Information Processing Standard

FQT — Formal Qualification Testing (DoD–STD–2167A)

FSD — Full-Scale Development (AFR 800–14)

FTP — File Transfer Protocol

FTR — Final Technical Report

FW — Firmware

GAO — Government Accounting Office (U.S. Government)

GEAE — General Electric Aircraft Engines

GECRD — General Electric Corporate Research and Development

GFE — Government Furnished Equipment (AFR 800–14)

GFP — Government Furnished Property (AFR 800–14)

GFS — Government Furnished Software (AFR 800–14)

GKS — Graphics Kernel System

GOSIP — Government Open Systems Interconnection Profile

GRC — General Research Corporation

GUI — Graphical User Interface

HDDBMS — Heterogeneous Distributed Database Management System

HDL — Hardware Description Language

HOL — High Order Language (DoD–STD–2167A)

HW — Hardware

HWCI — Hardware Configuration Item (DoD–STD–2167A)

I/O — Input/Output

IBM — International Business Machines

IC — Integrated Circuit

ICWG — Interface Control Working Group

IDA — Institute for Defense Analyses

IDD — Interface Design Document (DoD–STD–2167A)

IDL — Interface Description Language (Carnegie-Mellon University)

IEEE — Institute of Electrical and Electronics Engineers

IGES — Initial Graphics Exchange Specification (MIL–STD–28000)

IITRI — IIT Research Institute

IRDS — Information Resource Dictionary System

IRS — Interface Requirements Specification (DoD–STD–2167A)

ISA — Instruction Set Architecture

ISO — International Standards Organization

ISTAR — [SWV: DOW87] reference indicates that ISTAR was developed by Imperial Software Technology (IST). This may be the first three letters of the acronym.

IV&V — Independent Verification and Validation (DoD-STD-2167A)

JIAWG — Joint Integrated Avionics Working Group

JPO — Joint Program Office

JSSEE — Joint Services Software Engineering Environment (STARS)

K/O — Kick-Off

KAPSE — Kernel APSE

KBSA — Knowledge-Based Software Assistant (RL consortium)

KIT/KITIA — KAPSE Interface Team/KIT - Industry - Academia

LAN — Local Area Network

LOC — Lines of Code

LSQE — Language Sensitive Quality Editor (SPS)

MAPSE — Minimal APSE

MASA — Modular Avionics System Architectures (WPAFB)

MCC — Microelectronics and Computer Technology Corporation

MCCR — Mission Critical Computer Resources

MCCS — Mission Critical Computer Systems

MDC — McDonnell Douglas Corporation

MIL-STD — Military Standard

MOO — Menu Operation Organizer (SLCSE)

MS-DOS — Microsoft DOS

MULTICS — Multiplexed Information and Computing System

MVS — Multiple Virtual Storage (IBM)

NAC — Naval Avionics Center

NADC — Naval Air Development Center

NASA — National Aeronautics and Space Administration

NBS — National Bureau of Standards

NBS/ICST — National Bureau of Standards Institute for Computer Science and Technology

NCA — Network Computing Architecture (Apollo)

NCS — Network Computing System (HP's Apollo Systems Division)

NFS — Network File System

NIST — National Institute of Science and Technology (formerly NBS)

NRL — Naval Research Laboratory (Washington, D.C.)

NSE — Network Software Environment (Sun)

OFPP — Office of Federal Procurement Policy (U.S. Government)

OMB — Office of Management and Budget

OMG — Object Management Group Inc.

OODB — Object-Oriented Database

OODBMS — Object-Oriented Database Management System

OOP[S] — Object-Oriented Programming [System]

OOPL — Object-Oriented Programming Language

OPR — Office of Primary Responsibility (AFR 800–14)

OS — Operating System

OSD — Office of Secretary of Defense (U.S. Government)

OSF — Open Software Foundation

OSI — Open System Interconnection (ISO/ANSI standard)

OT&E — Operational Test & Evaluation (AFR 800–14)

PC — Personal Computer

PCR — Problem/Change Report (DoD–STD–2167A)

PCTE — Portable Common Tool Environment (CEC, ESPRIT)

PCTE+ — militarized PCTE

PDES — Product Data Exchange Specification

PERT — Program Evaluation and Review Technique

PHIGS — Programmer's Hierarchical Interface Graphics System

PM — Program Manager (AFR 800–14)

POSE — Picture Oriented Software Engineering (Computer Systems Advisors)

POSIX — Standard Portable Operating System Interface for Computer Environments

PPDFC — Production Planning Document Flow Chart (MDC)

PTI — Public Tool Interface (PCTE)

QA — Quality Assurance

QUES — QUality Evaluation System (SPS, RL)

R&D — Research and Development

RAA — Responsibility, Authority, and Accountability (MDCDAC)

RL — Rome Laboratory (Griffiss AFB)

RDT&E — Research Development Test and Evaluation (DoD)

RFP — Request For Proposal

RPC — Remote Procedure Call

RPI — Rensselaer Polytechnic Institute

SAA — Systems Application Architecture (IBM)

SAIS — [re: organization or standard?]

SBIR — Small Business Innovation Research

SDI — Strategic Defense Initiative (DoD)

SDIO — Strategic Defense Initiative Office

SDL — Schema Definition Language (SLCSE)

SDP — Software Development Plan (DoD–STD–2167A)

SDR — System Design Review (DoD–STD–2167A)

SECD — Systems Engineering Concept Demonstration

SEE — Software Engineering Environment

SEI — Software Engineering Institute (DoD)

SEL — Software Engineering Laboratory (RL)

SELC — Systems Engineering Life Cycle (MDC)

SETIG — Software Engineering Technical Interchange Group (MDC)

SGML — Standard Generalized Markup Language (ISO 8879)

SLCSE — Software Life Cycle Support Environment (RL)

SNA — Systems Network Architecture (IBM)

SON — Statement of Need

SOW — Statement of Work

SPC — Software Productivity Consortium

SPM — Systems Program Manager (AFR 800–14)

SPS — Software Productivity Solutions, Inc.

SQL — Structured Query Language (ANSI standard)

SRR — System Requirements Review (DoD–STD–2167A)

SRS — Software Requirements Specification (DoD–STD–2167A)

SSDD — System/Segment Design Document (DoD–STD–2167A)

SSE — Software Support Environment (NASA)

SSR — Software Specification Review (DoD–STD–2167A)

SSS — System/Segment Specification (DoD–STD–2167A)

STARS — Software Technology for Adaptable, Reliable Systems (DoD)

STD — Standard; Software Test Description (DoD–STD–2167A)

STP — Software Test Plan (DoD–STD–2167A)

STSC — Software Technology Support Center (AF)

SW — Software

TCP/IP — Transmission Control Protocol/Internetwork Protocol

TISSS — Tester Independent Software Support System (RL)

TOA — Task Order Agreement

TTCP — The Technical Cooperation Program

UI — User Interface

UIMS — User Interface Management System

USAF — United States Air Force

V&V — Verification and Validation

VAX — Virtual Address eXtension (DEC)

VHDL — Very High Speed Hardware Description Language

VHLL — Very High Level Language

VHSIC — Very High Speed Integrated Circuit

VLSI — Very Large Scale Integration

VMS — Virtual Memory System (DEC)

WBS — Work Breakdown Structure

WPAFB — Wright Patterson AFB

WVU — West Virginia University

WYSIWYG — What-You-See-Is-What-You-Get

☆U.S. GOVERNMENT PRINTING OFFICE: 1993-710-093-60012

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C^3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.